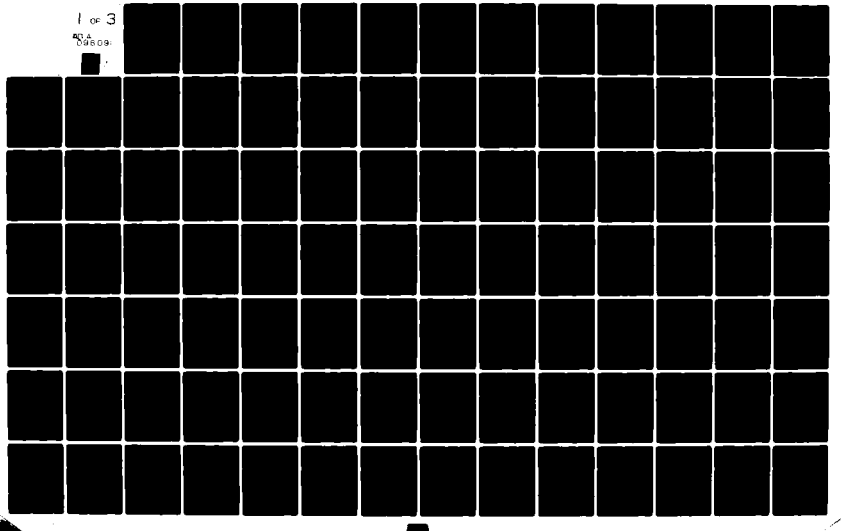
AD-A096 091 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2.1
DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR S--ETC(U)
DEC 80 R VISWANATHAN, W RUSSELL, A HIGGINS DCA100-79-C-0037
UNCLASSIFIED NL

1 OF 3
AD A
096091



Bolt Beranek and Newman Inc.

AD A096091

Report No. 4565

Design and Real-Time Implementation of a Robust APC Coder for Speech Transmission Over 16 Kb/s Noisy Channels

Volume I: Algorithm Design and Simulation

Final Report

December 1980

Prepared for:
Defense Communications Agency

Report No. 4565

DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR SPEECH TRANSMISSION OVER 16 Kb/s NOISY CHANNELS

Final Report

Volume I: Algorithm Design and Simulation

Authors: R. Viswanathan, W. Russell, A. Higgins

December 1980

Prepared for:
Defense Communications Agency

Unclassified.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN Report No. 4565, Vol. I	2. GOVT ACCESSION NO. AD-A096091	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR SPEECH TRANSMISSION OVER 16 Kb/s NOISY CHANNELS, Vol. I: Algorithm Design and Simulation		5. TYPE OF REPORT & PERIOD COVERED Final Report July 1979 - December 1980 6. PERFORMING ORG. REPORT NUMBER BBN Report No. 4565
7. AUTHOR(s) R. Viswanathan, W. Russell, and A. Higgins		8. CONTRACT OR GRANT NUMBER(s) DCA100-79-C-0037
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, MA 02238		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Agency Contract Management Division, Code 260 Washington, D.C. 20305		12. REPORT DATE December 1980 13. NUMBER OF PAGES 285
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

speech coding, adaptive predictive coding, mediumband speech coders, 16 kb/s speech transmission, noise spectral shaping, limit-cycle behavior, robust speech coder, digital voice terminal, real-time speech coder.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the design and development of a real-time robust APC speech coder that transmits high-quality speech over a 16 kb/s channel with bit-error rates of up to 1%. This volume (Vol. I) describes the development, experimental investigation, and speech-quality optimization of 16 kb/s adaptive predictive coders. Important among the various topics treated in this report are: extensive investigation of several existing and new methods for coding the APC residual, adaptive spectral shaping of the →

cont'd.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract cont'd.

quantization noise, causes for and effective remedy of the limit-cycle behavior of the quantizer output, different APC configurations, optimization of APC coders for error-free channels and for errorful channels, tradeoff between voice data rate and error-protection data rate, tandeming of the optimized 16 kb/s APC coder with the 2.4 kb/s LPC-10 coder, and coder performance in noisy backgrounds typical in offices and in air-borne command post environments. Most impressive of the numerous interesting and useful results described in this report is that the final optimized, robust coder design yields speech quality that degrades only slightly as the bit-error rate is increased from 0% to 1%.

ACKNOWLEDGMENTS

The work by J. Makhoul and M. Berouti on entropy-coded APC system with noise shaping served as the starting point for this project. The authors would like to thank G. Moran, the COTR of this project, for his interest and interaction on technical issues during the course of this project. The author (RV) would also like to thank B. Atal for valuable discussions on some aspects of his work on adaptive predictive coding of speech.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The goal of this project was the design and development of a real-time speech coding system that produces high quality speech at a data rate of 16 kb/s (kilobits/second). The final report of this project is organized in two volumes. Volume I, which is this report, describes our work on the development and optimization of the speech coding algorithm. Volume II deals with the implementation of the final optimized speech coding algorithm as a real-time full duplex system on a CSP Inc. MAP-300 signal processing computer and associated hardware.

In this chapter, we state the design requirements on the speech coder performance (Section 1.1), describe briefly the optimized coder (Section 1.2), and provide an overview of the rest of this report (Section 1.3).

1.1 Coder Design Requirements

The input speech of the coder should have a bandwidth of at least 3.2 kHz. The encoder and decoder of the speech coder should operate independently, with the encoder mapping the analog input signal into an output binary sequence and the decoder mapping the binary sequence into the corresponding analog output

speech. In addition to the requirement that the speech coder in general produce speech of very good quality in the sense that it has a very high degree of user acceptance, there are several specific requirements on the coder performance as given below:

1. Noisy channel: Produce good quality speech under conditions of a transmission bit error rate of up to 1%.

2. Acoustic background noise: Produce toll quality speech under conditions of acoustic background noise such as office noise with a sound pressure level (SPL) of 60 dB re 20 micronewtons per square meter, and good quality speech under 100 dB of acoustic background noise such as in Air-Borne Command Post (ABCP) environment.

3. Tandem operation with LPC-10 coder: Perform satisfactorily in tandem (in both directions) with an LPC-10 speech coder operating at a data rate of 2.4 kb/s. The tandem link should provide speech intelligibility with minimal degradation compared with a single link of the 2.4 kb/s LPC-10 coder alone.

Other objectives of this work have included:

1. Minimize the computational complexity of the speech coding algorithm.

2. Identify and explain the features in the optimized 16 kb/s coder and in the 2.4 kb/s LPC-10 coder that control the quality of the tandem link between the two coders, and indicate how the tandem performance could be further improved.

3. Extend the design of a previously developed 9.6 kb/s baseband LPC coder [3] to the 16 kb/s data rate, and compare the output speech quality of the resulting baseband coder with that of the optimized 16 kb/s coder.

1.2 Summary of the Optimized Algorithm

For the speech coding algorithm, we chose the adaptive predictive coder (APC). The optimized APC algorithm may be summarized as follows. In the transmitter, the analog input speech is lowpass filtered at 3.2 kHz, sampled at 6.621 kHz, and divided into frames of 32.625 ms duration. Each frame of speech is preemphasized using the filter $(1-0.4z^{-1})$ and encoded using the APC encoder, to produce the quantized residual samples for that frame. The APC encoder employs (1) 3-tap pitch prediction and 6-pole spectral prediction to obtain the residual, (2) forward-adaptive quantization of the residual, and (3) pole-zero

spectral shaping of the quantization noise to reduce its perception at the coder output. The parameters of the spectral and pitch predictors and of the adaptive quantizer are quantized, coded, partially error-protected, and transmitted along with the encoded residual samples.

In the receiver, the decoded residual samples are applied to the input of a cascade of the 6-pole spectrum synthesis and the 3-tap pitch synthesis filters. The output of the cascade is deemphasized using the filter $1/(1-0.4z^{-1})$, D/A converted, and lowpass filtered at 3.2 kHz to produce the analog output speech.

1.3 Overview of the Report

In Chapter 2, we provide the rationale for our choice of the APC coder for this work and of the 6.621 kHz sampling rate for its input speech, and we describe three input-speech data bases we employed during this work. Chapter 3 reviews briefly the details of the APC coder. In Chapter 4, we describe three types of APC coder configurations that we investigated in this work. In Chapter 5, we define an important quantity called the feedback gain of the APC transmitter and show how it is related to the various APC parameters. The feedback gain concept is used throughout the later chapters to link the occurrence of

undesirably large amounts of quantization noise with positive values of the feedback gain. In the next four chapters, we describe in detail our work on developing the various aspects of the APC system: quantization of APC parameters (Chapter 6); methods for adaptive noise shaping (Chapter 7); methods for coding the APC residual (Chapter 8); and methods for preventing the "limit-cycle" behavior of the APC system (Chapter 9). The results of our work on APC algorithm optimization at 16 kb/s are reported in Chapter 10 for error-free channels, and in Chapters 11 and 12 for noisy channels. Chapter 11 describes several optimized APC coders, each with bits allocated for error protection of parameters but operating over error-free channels, while Chapter 12 contains the results of evaluation of the performance of these optimized coders in 1% channel error. In Chapter 12, we also report a single APC system design as being the most robust and best overall 16 kb/s coder. The performance of this optimized coder in office noise and in ABCP noise environments is treated in Chapter 13, while its performance in tandem with LPC-10 is described in Chapter 14. In Chapter 15, we present a design of a 16 kb/s baseband coder as well as the comparative results of this coder and the optimized APC coder. In Chapter 16, we describe several modifications to the optimized coder to simplify some aspects of the coder and to make refinements to the coder design. Also in Chapter 16, we

summarize the details of the final optimized 16 kb/s APC coder and present the results of testing the coder's real-time implementation on the MAP-300. Finally, in Chapter 17, we summarize the results of this work, and identify explicitly what we believe are the major contributions of this work.

Contained in the appendices are: Specification of the Optimized 16 kb/s APC Algorithm (Appendix A); User's Guide for the FORTRAN Simulation of the Optimized 16 kb/s APC Coder (Appendix B); and a listing of the source programs of this FORTRAN simulation (Appendix C).

2. CHOICE OF 16 KB/S CODER

2.1 Rationale for Choosing APC

As candidates for our choice of the 16 kb/s speech coding algorithms, we considered a number of coders including adaptive residual coder, APC, delta modulation systems, sub-band coder, adaptive transform coder (ATC), and baseband coder (BBC). We investigated each of these coders to see if it could satisfy the requirements given in Section 1.1. We concluded that some of the coders cannot produce toll quality speech at 16 kb/s. The coders that are capable of transmitting toll quality speech at 16 kb/s (assuming good quality input speech and error-free channel) are APC, ATC, and BBC. Flanagan's recent assessment is in agreement with this conclusion [2].

Previous work at BBN has dealt with both APC [5] and BBC [3,4] systems. The results of this work show that the APC system with an appropriate noise spectral shaping produces output speech at 16 kb/s that is almost indistinguishable from noise-corrupted input speech with a signal-to-noise ratio of 10-30 dB [5]. The BBC coder produces either background roughness or low-level tones, depending on the method of high-frequency regeneration used [3,4]. For the ATC coder [6], proper decoding of the

transmitted signal data (transform coefficients) requires an error-free transmission of the side information. This indicates the strong possibility that the quality of the ATC speech would degrade drastically in a relatively high channel-error environment. In contrast, an error in the side information of the APC coder may change the spectral envelope and cause perceivable distortion, but the degradation may be much more graceful than might happen in ATC. Therefore, we chose APC as the best overall approach to the present application.

2.2 Sampling Rate of Input Speech

One of the requirements on the speech coder is that the bandwidth of the input speech of the coder be greater than or equal to 3.2 kHz. The audio signal interface provided by GTE Sylvania for the MAP-300 array processor provides lowpass filters with -3 dB cutoffs of 3.2 kHz and 3.8 kHz (Appendix A in [3]). Therefore, the input sampling rate FS may be chosen to have a value around 6.67 kHz or 8 kHz. Since the coding and error-protection of the side information of the APC coder is expected to take up about 3 kb/s, choosing FS=8 kHz leads to a residual quantization accuracy of only about 1.6 bits/sample. The resulting quantization noise may more than offset the advantage that the choice of FS=8 kHz may yield slightly higher

intelligibility. Also, the computational load is greater with the higher sampling rate. Therefore, we chose FS=6.67 kHz as the approximate sampling rate of the input speech.

The exact value of the sampling rate has to be selected from the options provided by the real-time clock in the audio signal interface for the MAP-300 [3]. The primitive clock rate provided by the master oscillator within the interface is 384 kHz. A candidate sampling rate is given by 384/D, where D is the programmable, integer divide-ratio. We chose the sampling rate of 384/58 (approximately 6.621 kHz), since this choice 1) avoids aliasing and 2) yields a variety of 16 kb/s coder realizations with different frame sizes and having integer numbers of both samples per frame and bits per frame. In all the simulations of the APC coders on our PDP-10 computer, we used a sampling rate of 6.67 kHz (or 150-microsecond sampling period), since it is close to the chosen sampling rate and since all the simulation results can be simply carried over to the real-time system. For example, a frame size of 27 ms contains 180 speech samples at 6.67 kHz; the corresponding frame size for the real-time system is 27.1875 ms, which also has 180 speech samples.

2.3 Data Bases

We employed three data bases of 11-bit linear PCM speech in this project: a high-quality data base, an "office-noise" data base, and an ABCP data base. The high-quality data base has 12 sentences of about 2-3 seconds duration each, with equal numbers of sentences from male and female talkers. This data base is the same as the one used in a previous DCA contract at BBN [3]. The signal-to-noise ratio of the speech in this data base is about 60 dB. The office-noise data base has 10 sentences, which we digitized at 6.67 kHz directly from a sponsor-supplied audio tape recorded in an office-noise environment (with the acoustic background noise at a level of about 60 dB SPL re 20 micronewtons per square meter). For the ABCP data base, we digitized a number of utterances from a sponsor-supplied audio tape containing speech recorded in an ABCP environment. The level of the background acoustic noise in such an environment is typically about 90 dB SPL.

3. REVIEW OF THE APC SYSTEM

3.1 Basic APC system.

The basic APC system is depicted in Fig. 1. The feedback structure, which constitutes the transmitter, encodes the sampled input speech $S(z)$[1] in terms of the quantized residual $\hat{W}(z)$ and the spectral (or linear prediction) and pitch inverse filters $A(z)$ and $C(z)$ given by:

$$A(z) = 1 + \sum_{k=1}^{p} a(k) z^{-k}, \tag{1}$$

and

$$C(z) = 1 + \sum_{k=M-m}^{M+m} c(k) z^{-k}, \tag{2}$$

where $a(k)$, $1 \leq k \leq p$, are the spectral predictor coefficients; $c(k)$, $M-m \leq k \leq M+m$, are the pitch predictor coefficients; and M is the pitch period in number of samples. We refer to the order of the spectral predictor p as the LPC order and the order of the pitch predictor $2m+1$ as the number of pitch-filter taps. The spectral predictor $A(z)$ is designed to remove the redundancy due to

---

[1] $S(z)$ denotes the z-transform of the time signal $s(n)$.

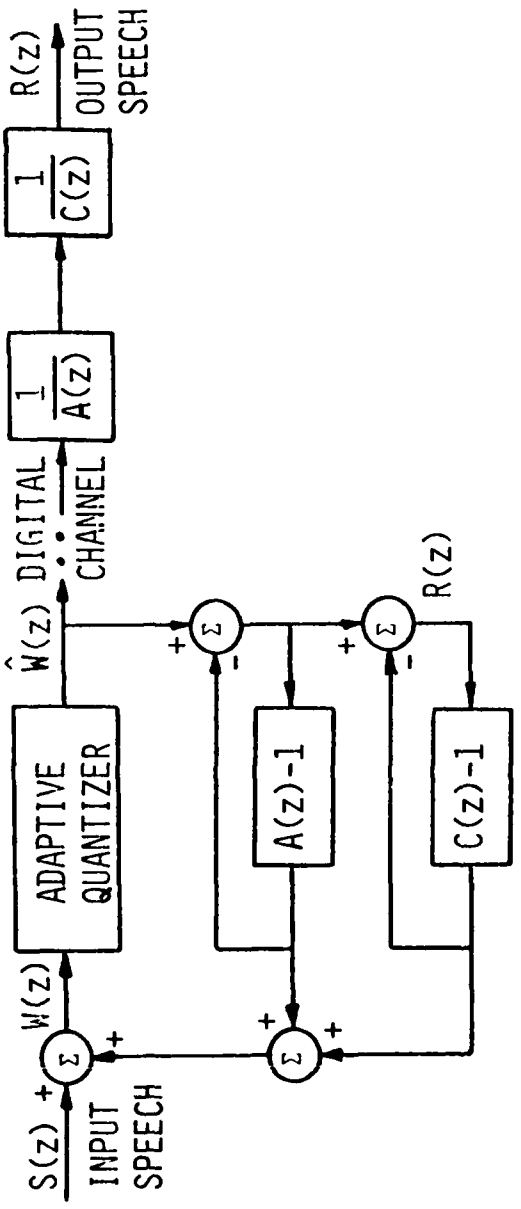


FIG. 1. Basic APC system.

spectral or short-term correlations, while the pitch predictor C(z) removes the redundancy due to long-term correlations produced by pitch periodicity. The early implementations of APC [7-10] have either used no pitch predictor (c(k)=0, for all k) or used a 1-tap predictor (m=0). Recently, a 3-tap predictor (m=1) has been used [11]. The APC residual W(z) is quantized into $\hat{W}(z)$, which is transmitted across the digital channel. At the receiver, the decoded signal is filtered by the all-pole spectral filter 1/A(z) and then by the pitch-synthesis filter 1/C(z), to produce the output speech R(z). In the absence of channel bit-errors, it can be shown that

$$R(z) = S(z) + Q(z), \tag{3}$$

where Q(z) is the quantization noise:

$$Q(z) = \hat{W}(z) - W(z). \tag{4}$$

The spectral and pitch predictor coefficients and the parameters of the quantizer are varied adaptively in time to track the changing properties of the input speech signal. There are two types of adaptive schemes: (1) forward-adaptive schemes, which transmit the parameters being adapted, once every frame [8,9]; and (2) backward-adaptive schemes, which do not transmit any parameters and estimate them at the receiver from the decoded residual samples [8,12]. Since the performance of the backward-

adaptive schemes would degrade significantly at channel-error rates as high as 1%, we chose to consider only forward-adaptive schemes.

The parameters of the predictors A(z) and C(z) are computed in such a way that the mean-square value of the quantization noise Q(z) is minimized. In our simulation of the APC system in Fig. 1, the pitch and the taps are computed from the speech signal using either the autocorrelation method or the covariance method of linear prediction [13] (see Section 8.2.1); the speech signal is then inverse-filtered with C(z) to produce the "first residual" E1(z); the spectral coefficients a(k) are computed from E1(z) using the autocorrelation linear prediction method; the residual signal E1(z) is inverse-filtered with A(z) to produce the "second residual" E2(z); and finally, the parameter(s) of the adaptive quantizer are computed from this second residual (see Section 3.2 and Chapter 8 for more details on the adaptive quantizer). The two inverse-filtering operations just mentioned and the two prediction operations A(z)-1 and C(z)-1 within the feedback structure in Fig. 1 are performed using the corresponding quantized parameters, to correspond to what the receiver does; this also yields a smaller mean-square value for the quantization noise than if we had used unquantized parameters. Similarly, the parameter(s) of the quantizer are also quantized before being used in the APC loop.

3.2 Forward-Adaptive Residual Quantization

The adaptive quantizer that we have used in this work is shown in Fig. 2. This quantizer has a gain normalization 1/G followed by an optimum (uniform or nonuniform) unit-variance quantizer. Ideally, the value of G is chosen such that the normalized APC residual $\hat{U}(z)$ has unit variance. Since the APC residual W(z) becomes available only as the APC loop is in operation, G is computed approximately as the rms value of the second residual E2(z). G is transmitted to the receiver along with the encoded $\hat{U}(z)$. $\hat{W}(z)$ is computed from $\hat{U}(z)$, both at the transmitter and at the receiver, by multiplying it with G. The unit-variance quantizer is usually designed by minimizing the mean-square error between U(z) and $\hat{U}(z)$, assuming a certain probability distribution for U(z) (e.g., Gaussian, Laplacian, gamma, etc.) [14,15].

The residual quantizer produces two kinds of quantization error: 1) clipping error, which is produced whenever the value of signal u(n) lies outside the extreme ranges of the quantizer; and 2) roundoff error or granular noise, which is produced whenever u(n) lies within the extreme ranges of the quantizer. Granular noise, which causes a degradation in the output speech in the form of broad-band background noise, usually constitutes the

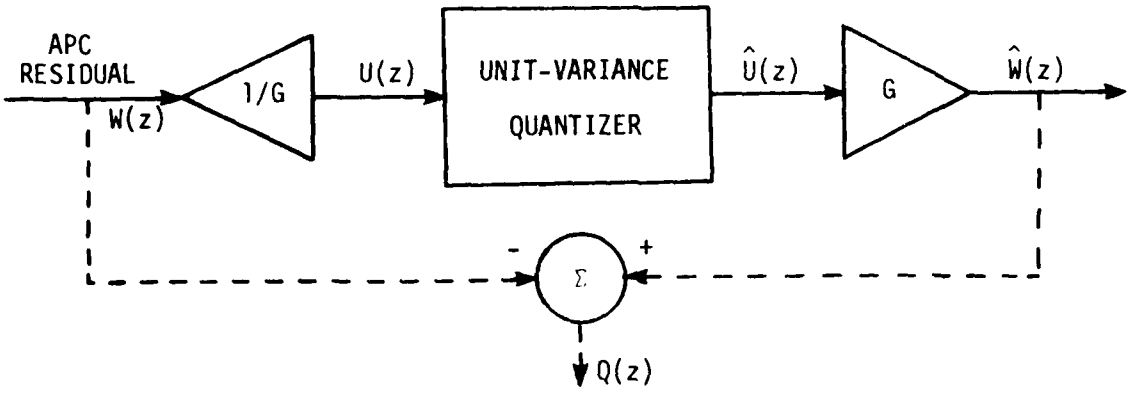


FIG. 2. Forward-adaptive quantizer for the APC residual.

dominant form of degradation. Clipping errors, on the other hand, cause undesirable degradation in the form of "pops" or "clicks"; such effects can be perceived even when the incidence of clipping errors is as low as 0.1% [5]. In Chapter 8, we present several methods for reducing the clipping errors. In the next subsection, we review an approach to reduce the perception of the granular noise.

3.3 Noise Spectral Shaping

If the adaptive quantizer in Figs. 1 and 2 is designed such that the quantization noise Q(z) is mostly granular noise, then Q(z) has a flat spectral envelope, which can mask the speech spectrum at high frequencies; this causes the perception of a hissing background noise in the output speech R(z) [5,11]. To minimize the perception of such noise, proposals have been made recently for proper shaping of the noise spectrum [5,11], resulting in the following revised expression for R(z):

$$R(z) = S(z) + B(z)Q(z); \tag{5}$$

the filter B(z) is designed to shape the noise spectrum in a way that yields a perceptually more pleasing output speech. Let us denote the new output noise as $Q'(z)$

$$Q'(z) = B(z)Q(z). \tag{6}$$

The specific noise shaping methods that we investigated and their implementation issues are treated in Chapter 7.

3.4 Signal-to-Noise Ratio Considerations

The signal-to-output noise ratio in dB is denoted by S/Q'. This may be computed in one of two ways: long-term method or segmental (short-term) method. The long-term S/Q' is computed

from the energies of S(z) and Q′(z) calculated over a long duration (e.g., over individual sentences). The segmental S/Q′ is computed as the average over frames (typically 25 ms long) of the frame-based ratio in dB [16]. The segmental S/Q′ ratio has been found to correlate better with subjective perceptual judgments than the long-term S/Q′ ratio [17,18]. Although we computed both ratios in our simulations, we shall give only the segmental S/Q′ values, and we shall also drop the qualifier "segmental" and the prime in S/Q′, for convenience.

We make two observations based on the results of our experimental work. First, the S/Q ratio overestimates the effect of clipping errors introduced by the quantizer. That is, a coder with clipping errors will have a significantly lower S/Q ratio than another coder with only granular noise, notwithstanding that the first coder may in fact have similar or better speech quality. Second, the spectral shaping of the quantization noise Q(z) reduces the S/Q ratio but enhances the perceived speech quality. Both observations should caution the reader not to take S/Q ratios, given in this report or elsewhere, as strictly indicative of perceived speech quality.

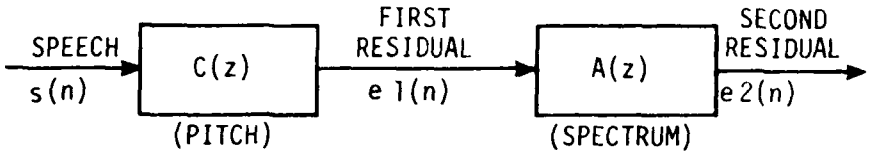
4. DIFFERENT APC CONFIGURATIONS

In this chapter, we present two methods of sequencing the spectral and pitch predictors and three types of configurations for the APC system.

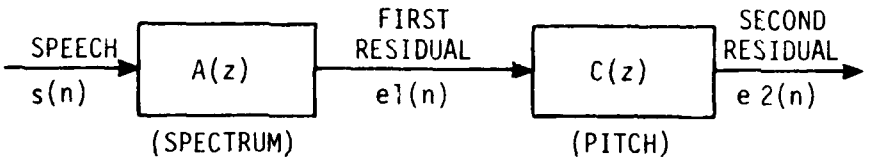
4.1 Sequencing of Spectral and Pitch Predictors

For the APC coder in Fig. 1, we have assumed the prediction sequence of pitch-followed-by-spectrum (denoted by P-S) in the sense that this sequence indicates the order, as discussed in Section 3.1 and as shown in Fig. 3(a), in which the parameters of the pitch and the spectral predictors are estimated from the speech signal. The order of predictors, C followed by A in this case, must be employed in a consistent manner in estimation, encoding (within the APC loop), and synthesis. Inconsistency in the ordering has been found to cause perceivable distortions ([5]; also see Section 12.5.2).

The first APC work employed the P-S sequence [7]; most APC implementations thus far have used this sequence as well. Only recently has the S-P (spectrum-followed-by-pitch) prediction sequence been considered [11]. The APC system using the S-P prediction sequence is obtained from Fig. 1 by simply



(a) Pitch-followed-by-spectrum prediction sequence



(b) Spectrum-followed-by-pitch prediction sequence

FIG. 3. Two methods of sequencing the spectral and pitch predictors.

interchanging the symbols A(z) and C(z). The order of estimation of the predictor parameters in this case is shown in Fig. 3(b).

From a mathematical point of view, since the minimum-mean-square-error estimation of the spectral predictor parameters results in large residual amplitudes at the pitch pulses, it may be argued that removing the pitch redundancy first should improve the subsequent spectral prediction. This viewpoint supports the use of the P-S prediction sequence. On the other hand, since the synthesis for the S-P sequence performs the pitch reconstruction first and then the spectral shaping function, the S-P sequence corresponds to the way the human speech production physically happens. These considerations by themselves do not suggest as to which prediction sequence should be used. Nonetheless, the prediction sequence, as will be seen in the later chapters, plays an important role in the design of the APC coder.

4.2 Prediction-Feedback Configuration

Figure 4 shows the APC configuration that we call the prediction-feedback (PF) configuration, or APC-PF. This configuration is the same as the one in Fig. 1, except that Fig. 4 includes noise shaping as discussed in Section 3.3. Notice that the feedback structure in Fig. 4 employs both A(z) and C(z)

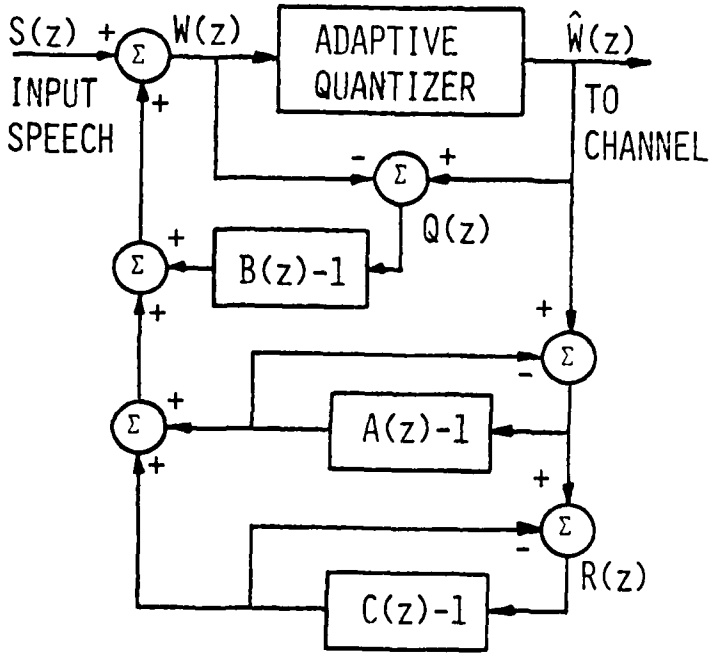


FIG. 4. The prediction-feedback configuration of APC. The figure shows the transmitter only. The receiver contains the filter cascade [1/A(z)] [1/C(z)] as shown in FIG. 1.

in a predictive manner, as follows. The output of the filter [C(z)-1] is the predicted value of the output R(z) using the pitch prediction. Since the input to the filter [A(z)-1] is the receiver's first residual $E1'(z)=C(z)R(z)$, its output is the predicted value of $E1'(z)$ using the spectral prediction. The APC-PF configuration was originally proposed in [5].

4.3 Noise-Feedback Configuration

The APC-NF (NF for noise feedback) configuration is shown in Fig. 5. This configuration was originally proposed in [19] and was used later in [20-22]. Below, we make three observations. First, we observe that in the APC-NF configuration, the input to the feedback structure is the second residual E2(z):

$$E2(z) = A(z)E1(z) = A(z)C(z)S(z). \tag{7}$$

Second, only the quantization noise Q(z) is fed back to the input. That is, there is no feedback path from the quantized residual $\hat{W}(z)$ as in the APC-PF configuration. The third observation that follows is quite important. The feedback transfer function F1(z), which is given by

$$F1(z) = A(z)C(z)B(z) - 1, \tag{8}$$

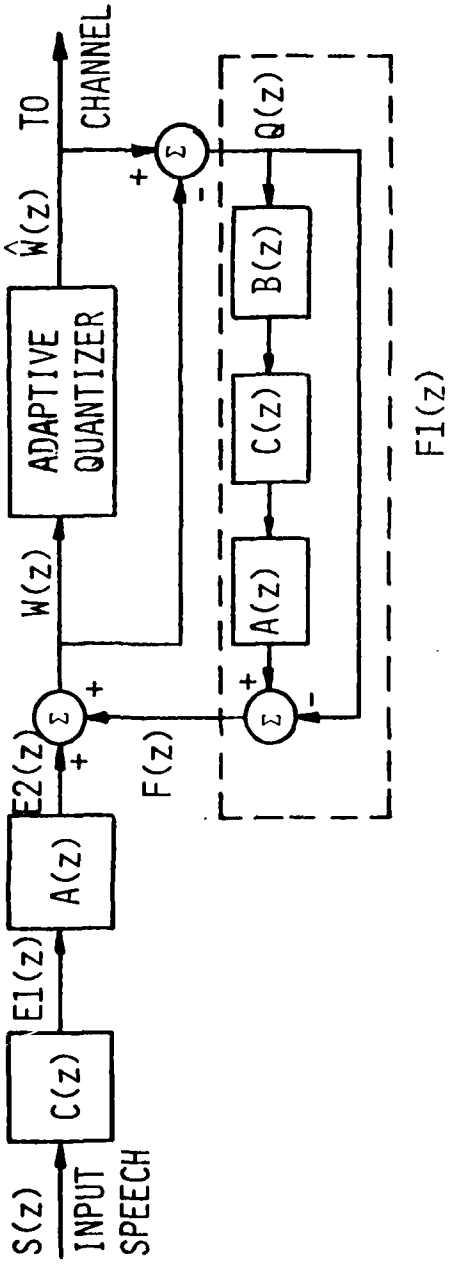


FIG. 5. The noise-feedback configuration of APC. The figure shows the transmitter only. The receiver contains the filter cascade [1/A(z)] [1/C(z)] as shown in FIG. 1.

can be implemented as a single filter by multiplying out the individual filter functions. But, as simple calculations will show, this procedure requires additional storage and/or computations. A simpler procedure is to implement them as a sequence of 3 filters along with a straight feedforward branch with a transfer gain of -1, as shown in Fig. 5. But, in what order should the 3 filters occur? Does it matter? Yes, it matters. Any order other than the one shown in Fig. 5 will produce, as we have experimentally found, a significant drop in S/Q ratio and may produce occasional "squeals" in the output speech. In one experiment, the S/Q ratio dropped from 18 dB to 16 dB when we switched the filter sequence from BCA to BAC. This non-commutativity is the result of the frame-by-frame time variation of the filters involved. The correct filter sequence can be obtained starting with the APC-PF case and deriving from it the APC-NF case, either by carefully keeping track of the orders of z-transformed quantities or through a series of straightforward block-diagram manipulations.

4.4 Hybrid-Feedback Configuration

An example of the APC-HF (HF for hybrid feedback) configuration is shown in Fig. 6. For this example, which was proposed in [11], the pitch predictor C(z) is placed in a

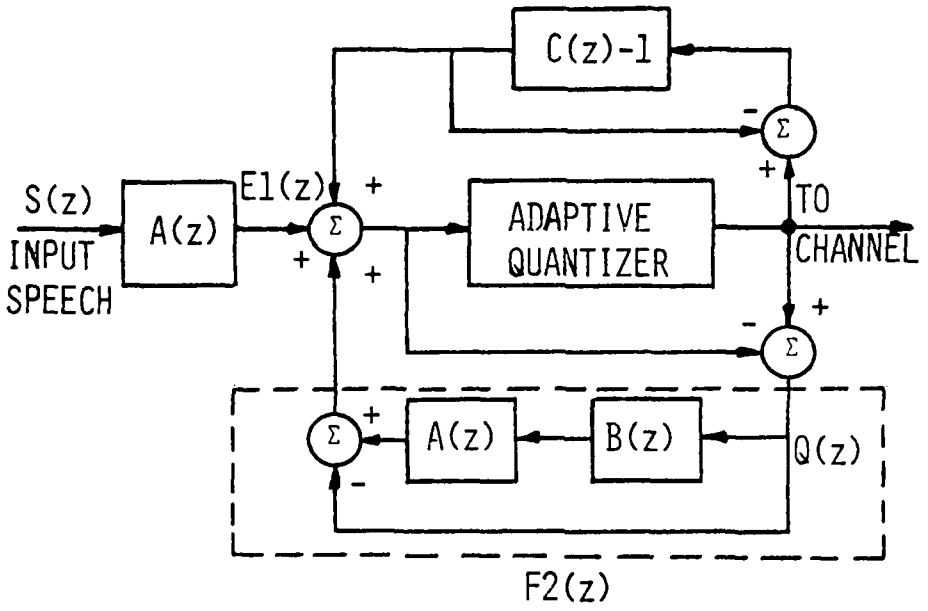


FIG. 6. The hybrid-feedback configuration of APC. The figure shows the transmitter only. The receiver contains the filter cascade [1/C(z)] [1/A(z)].

predictive manner while the spectral predictor A(z) is placed along with B(z) in a noise-feedback manner. One can verify that the correct filter sequence in the NF path is B(z)A(z), as given in the figure. It may also be seen that the example shown in Fig. 6 employs the S-P prediction sequence.

4.5 Comparison of the APC Configurations

Let us summarize the two conditions under which any two configurations become equivalent to each other: 1) same prediction sequence for the two configurations, which is used consistently in estimation, encoding, and synthesis; and 2) correct order of the filters A(z), B(z), and C(z) within the APC encoder. For example, the APC-PF configuration that is equivalent to the APC-HF configuration given in Fig. 6 is obtained by interchanging the symbols A(z) and C(z) in Fig. 4.

The relative properties of the three configurations (and the two prediction sequences) are discussed in later chapters with respect to 1) ease of implementation of noise shaping (Chapter 7) and 2) insertion of a limiter in the path of the quantization noise Q(z) in an attempt to prevent the build-up of excessive quantization noise (Chapter 9). The concept of feedback gain of APC is introduced in the next section, using the noise-feedback configuration.

5. FEEDBACK GAIN OF APC

Referring to the APC-NF configuration in Fig. 5, the APC residual W(z) is given by

$$W(z) = E2(z) + F(z). \quad (9)$$

Thus, the APC residual is the sum of two components: the second residual E2(z) and the quantization noise Q(z) filtered by F1(z) = A(z)C(z)B(z) - 1. If the energy of F(z) exceeds the energy of E2(z) for any frame, then the APC residual contains less speech information than quantization noise. Carrying this argument further, if the energy of W(z) is equal to the energy of the filtered noise, then W(z) becomes totally dominated by noise, and the output R(z) becomes non-speech and is usually perceived as "glitches" or "beeps."

To formalize these observations, we define the feedback gain of APC, G_F as the F/W ratio in dB. This definition has the flavor of the definition of the loop gain found in classical control theory texts, where the signal F(z) = F1(z)Q(z) is referred to as the return signal. Our experiments have shown that when the feedback gain G_F is positive for a frame, the quantization noise builds up to excessive values due to frequent clipping errors, leading to a "limit-cycle" behavior of the

quantized signal $\hat{W}(z)$ and producing glitches or beeps in the output speech. In one experiment, we found that the supposedly unit-variance signal $\hat{U}(z)$ took on large values ranging between -20 and 40, and that the quantizer output levels exhibited a limit-cycle behavior, banging between the two extreme levels.

Below, we derive an expression for the feedback gain G_F in terms of the contributions from the quantizer and from the filter Fl(z). Clearly,

$$G_F = F/W = F/Q - W/Q, \tag{10}$$

where the W/Q ratio is the signal-to-quantization-noise ratio of the adaptive quantizer in dB. If we assumed that the quantization noise is uncorrelated, then it can be shown that the F/Q ratio is the power gain of the function Fl(z); this power gain is denoted by $G_p(Fl)$ and given by the sum expressed in dB of the squares of the coefficients of powers of z^{-1} in the function Fl(z). Therefore,

$$G_F = G_p(Fl) - W/Q. \tag{11}$$

From the expression (11) and the results stated above, a power gain $G_p(Fl)$ larger than the W/Q ratio will lead to a limit-cycle behavior of the quantizer output $\hat{W}(z)$ and produce non-speech

output R(z). Notice that the APC system is always stable in the "bounded-input-bounded-output" sense. This is because of the clipping or saturation nonlinearity of the quantizer.

Finally, we make two remarks. First, although we introduced the feedback gain and derived the expression (11) for it using the APC-NF configuration, the conclusions given above are valid for the other two configurations as well. Second, for the purpose of investigating ways of reducing the power gain G_p(Fl) or G_p(ABC-1), one can consider the power gain G_p(ABC).

6. QUANTIZATION OF APC PARAMETERS

6.1 Spectral Parameters

After the spectral parameters a(k), $1 \leq k \leq p$, are computed, they are to be quantized for transmission to the receiver as well as for use in inverse filtering and APC encoding, as mentioned in Section 3.1. Our previous work has shown that optimal quantization of the spectral parameters can be accomplished by uniformly quantizing log area ratios (LARs), which are obtained by first converting predictor coefficients a(i) to reflection coefficients K(i) and then using the following logarithmic transformation [23,24]:

$$g(i) = 10 \log [1+K(i)]/[1-K(i)], \quad 1 \leq i \leq p. \qquad (12)$$

In most of our investigations, we used $p \leq 8$. The ranges in dB of the 8 LARs obtained for the high-quality data base are given in Table 1 for the case when the spectral parameters are extracted directly from the unpreemphasized speech signal. We used the ranges in Table 1 whenever the S-P prediction sequence was used, or whenever pitch prediction was not used. Table 2 gives the LAR ranges for the P-S sequence. Again, no preemphasis was applied to the input speech. (The LAR ranges for the preemphasized case are given in Table 12, Section 16.3.)

Coeff.#	# of Bits	Minimum (dB)	Maximum (dB)	Step size (dB)
1	6	-28.039	8.961	0.5781
2	5	-7.570	20.930	0.8906
3	4	-10.359	9.141	1.2187
4	4	-5.062	12.937	1.1250
5	4	-7.437	6.563	0.8750
6	4	-5.484	8.016	0.8438
7	3	-8.250	3.750	1.5000
8	3	-4.812	6.187	1.3750

TABLE 1. Quantization of the LARs of an APC system that uses the S-P prediction sequence and no preemphasis.

Coeff.#	# of Bits	Minimum (dB)	Maximum (dB)	Step size (dB)
1	6	-23.816	9.684	0.5234
2	5	-7.547	15.453	0.7188
3	4	-9.500	6.500	1.0000
4	4	-4.219	10.781	0.9375
5	4	-6.563	7.437	0.8750
6	4	-3.516	8.984	0.7813
7	3	-7.031	5.469	1.5625
8	3	-4.812	6.187	1.3750

TABLE 2. Quantization of the LARs of an APC system that uses the P-S prediction sequence and no preemphasis.

We employed a total of 33 bits and optimally allocated them among the 8 LARs using a method reported in [24]. The optimal bit allocation and step sizes for the two cases described above are also given in Tables 1 and 2.

6.2 Pitch Parameters

The pitch parameters are the pitch period M and the taps c(k), $M-m \leq k \leq M+m$. For the pitch frequency range 50-450 Hz that we assumed and the 6.67 kHz sampling rate, the pitch period M takes values in the range 14-133 samples, a total of 120 values. We used 7 bits to represent the pitch; therefore, M was "quantized" without error.

Considering the quantization of pitch taps, we investigated in this work the three cases: 1-tap (m=0), 3-tap (m=1), and 5-tap (m=2). We quantized the taps linearly using 4 bits for the center tap c(M) and 3 bits each for all the other taps. The ranges for the taps that we used in the quantization are given in Table 3 for the two methods of computing the taps: the autocorrelation method and the covariance method.

Tap ID	Autocorrelation Method		Covariance Method	
	Minimum	Maximum	Minimum	Maximum
Center tap	-0.93	0.10	-0.99	0.50
Other taps	-0.66	0.30	-0.90	0.35

TABLE 3. Quantization ranges for the pitch taps.

6.3 Residual Quantizer Gain

The gain parameter G, defined in Section 3.2, of the residual quantizer is quantized logarithmically. Without pitch prediction, G was found to take values from -5 dB to 45 dB. With pitch prediction, we used the range -10 to 46 dB. With the exception of the entropy coding system (see Section 8.4.2), we used 6 bits for quantizing the gain in all our experimental work; for the entropy coding system, we used 10 bits.

7. ADAPTIVE NOISE SHAPING METHODS

In this chapter, we describe the methods that we used for adaptive shaping of the quantization-noise spectrum. As a useful notation, we define

$$A(z/\alpha) = 1 + \sum_{k=1}^{p} a(k)\alpha^k z^{-k}, \qquad (13)$$

where $0 \leq \alpha < 1$; α may be expressed in terms of a bandwidth parameter w:

$$\alpha = \exp(-\pi w T), \qquad (14)$$

where T is the sampling interval. We note that the zeros of $A(z/\alpha)$ have the same frequencies as the zeros of A(z) but have bandwidths larger by w Hz. We observe that each of the noise-shaping methods described below has the property that it simultaneously reduces the S/Q ratio and the perception of the granular noise in the output speech.

7.1 All-Zero Noise Shaping

This method was proposed in [5] for the APC-PF system without pitch prediction. In this method, B(z) is an all-zero or FIR filter:

$$B(z) = 1 + \sum_{k=1}^{q} b(k)z^{-k}, \qquad (15)$$

where the leading term is unity so that the filter B(z)-1 is realizable within the APC loop (see Fig. 4). For the APC system without pitch prediction (any of the configurations in Figs. 4-6), it can be shown that

$$W(z) = A(z)S(z) + (A(z)B(z)-1)Q(z). \qquad (16)$$

Reference [5] suggests that B(z) be computed as the optimal inverse filter to A(z). While this noise shaping method reduces the perception of the granular noise in the output speech, we note that the particular criterion used for choosing B(z) produces also a second beneficial effect. This effect may be explained in two ways. First, from the second term on the right hand side of (16), we see that the above-mentioned criterion minimizes the noise contribution to the APC residual. Second, computing B(z) as the optimal inverse to A(z) is the same as minimizing the power gain of the filter-product A(z)B(z), for a given order q of B(z). From the discussions given in Chapter 5, the minimum-power-gain choice of B(z) will reduce the extent of the limit-cycle problem.

From our experiments, we found the choice q=1 (1-zero

shaping) to produce the best perceptual results. For this choice, the coefficient b(1) is computed as follows:

$$b(1) = -\rho(1)/\rho(0), \tag{17}$$

where

$$\rho(0) = 1 + \sum_{k=1}^{p} a^2(k), \tag{18}$$

and

$$\rho(1) = \sum_{k=0}^{p-1} a(k)a(k+1), \tag{19}$$

with a(0)=1.

For the APC system with pitch prediction, the equation (16) becomes

$$W(z) = A(z)C(z)S(z) + [A(z)C(z)B(z)-1]Q(z). \tag{20}$$

This last equation suggests that B(z) may be chosen by minimizing the power gain of A(z)B(z)C(z) (or as the optimal inverse to A(z)C(z)). For the 1-tap pitch prediction and for the 1-zero noise shaping, the minimization procedure yields a different coefficient $\hat{b}(1)$:

$$\hat{b}(1) = b(1)(1+c)/(1+c^2), \tag{21}$$

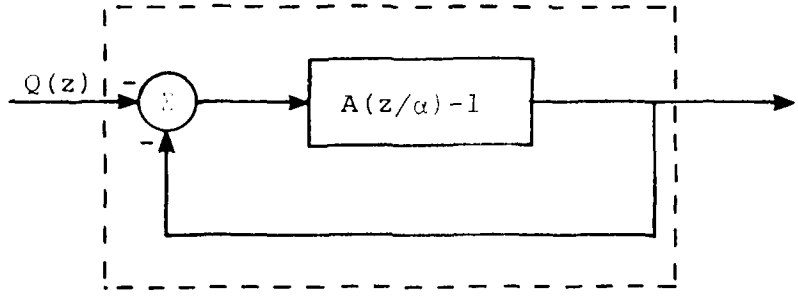
where b(1) is given by (17), and c is the pitch tap c(M). For c close to zero or one, $\hat{b}(1)$ is approximately equal to b(1). A brief experimental comparison of the two ways of computing b(1) showed that using $\hat{b}$ caused slightly additional roughness in the output speech. In all our subsequent work, we used equations (17)-(19) to compute the coefficient of the 1-zero filter.

7.2 All-Pole Noise Shaping

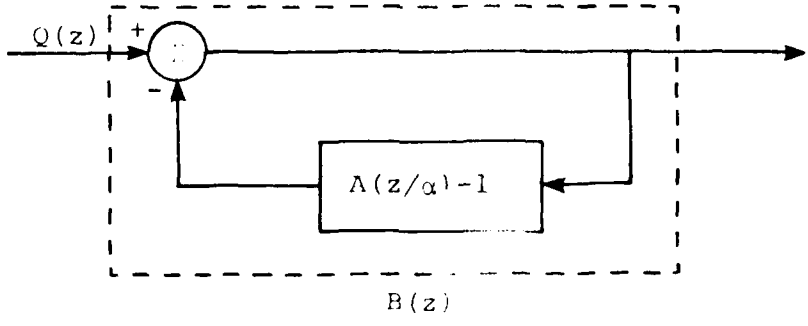
In this method, we set

$$B(z) = 1/A(z/\alpha), \tag{22}$$

where $A(z/\alpha)$ is given by (13). For this choice of B(z), we note that the smaller the value of the bandwidth parameter w, the larger the extent of the noise shaping and the smaller the resulting S/Q ratio. For this method, implementation of B(z)-1 for the APC-PF system is shown in Fig. 7(a), and implementation of B(z) for the APC-NF and the APC-HF systems is shown in Fig. 7(b). It can be shown that this all-pole noise shaping method also reduces the power gain of A(z)B(z) relative to that of A(z).



(a) Implementation of B(z)-1, for the APC-PF system.



(b) Implementation of B(z), for the APC-NF and APC-HF systems.

FIG. 7. Implementation of the all-pole noise-shaping with $B(z) = 1/A(z/\alpha)$.

7.3 Pole-Zero Noise Shaping

In this method, B(z) is a pole-zero filter. Following reference [11], we considered the special pole-zero filter

$$B(z) = A(z/\alpha)/A(z). \tag{23}$$

This choice of B(z) yields the following identity:

$$A(z)B(z) = A(z/\alpha), \tag{24}$$

which has two important consequences. First, defining $g_p(A)$ as the power gain in amplitude (rather than in dB) of the filter A(z), we have:

$$g_p(A) = 1 + \sum_{k=1}^{p} [a(k)]^2, \tag{25}$$

$$g_p(AB) = g_p(A(z/\alpha)) = 1 + \sum_{k=1}^{p} [a(k)\alpha^k]^2. \tag{26}$$

Since $0 \leq \alpha < 1$, equations (25) and (26) clearly show that the power gain of AB is less than the power gain of A. Second, a configuration in which A(z) and B(z) occur next to each other in a cascade is best suited for implementing the above pole-zero method, since the cascade reduces to the single filter $A(z/\alpha)$. We have two such configurations: APC-HF in Fig. 6, and APC-NF in

Fig. 5 but with the S-P prediction sequence. For configurations other than these two, pole-zero noise shaping can be implemented as shown in Fig. 8. Finally, for the choice of B(z) in (23), we note that the bigger the value of the bandwidth parameter w, the larger the extent of noise shaping and the smaller the resulting S/Q ratio.

7.4 Effect of Preemphasis on Noise Shaping

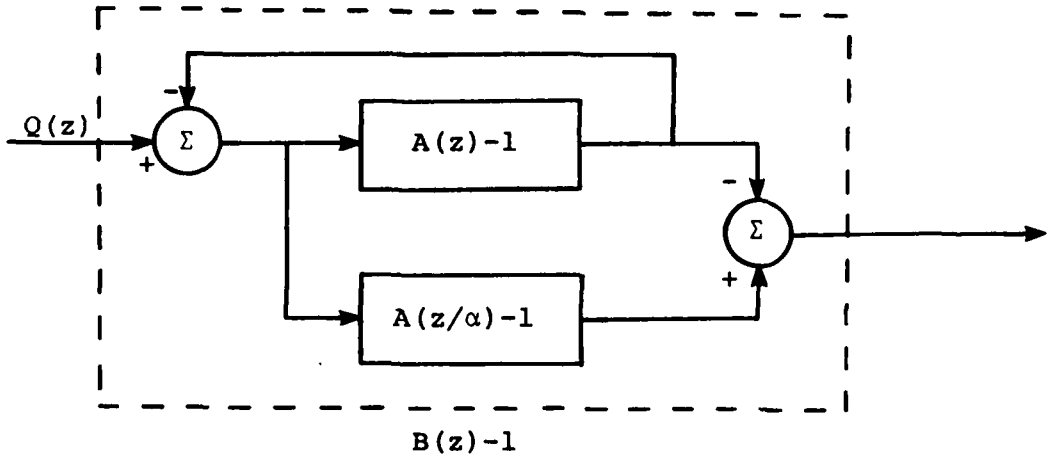
In Section 9.5, we discuss the use of preemphasis of the input speech with a filter

$$P(z) = 1 - \beta z^{-1}, \quad 0<\beta<1, \tag{27}$$

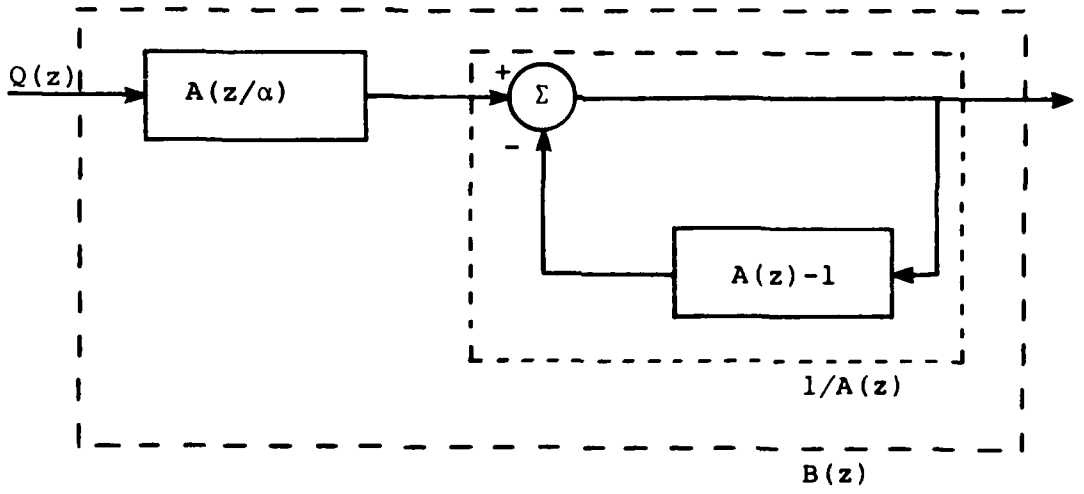
and the associated deemphasis of the coder output with $1/P(z)$. Here, we consider the effect of using preemphasis on a particular noise shaping method. For the APC coder using preemphasis, we can show that the output speech R(z) is given by

$$R(z) = S(z) + [B(z)/P(z)]Q(z). \tag{28}$$

Thus, the quantization noise is further shaped by the all-pole filter $1/P(z)$. This additional shaping decreases the quantization noise at high frequencies at the expense of increasing it at low frequencies. For the case where there is no noise shaping ($B(z)=1$), preemphasis reduced the perception of the



(a) Implementation of B(z)-1, for the APC-PF system.



(b) Implementation of B(z), for the APC-NF and APC-HF systems using the P-S prediction sequence.

FIG. 8. Implementation of the pole-zero noise shaping with $B(z) = A(z/\alpha)/A(z)$.

background noise but caused low-frequency roughness in the output speech. When we combined preemphasis with noise shaping, we found that the 1-zero method suffered the most, in that the output speech had perceivable roughness. The all-pole method was affected only slightly, but the pole-zero method was not affected in any perceivable manner.

7.5 Comparative Evaluation and Experimental Results

Figure 9 compares the different quantization noise spectra for a typical voiced sound. In this figure, plot (a) is the spectral envelope of the input speech; (b) is the unshaped (i.e., B(z)=1) noise spectrum; and (c) and (d) correspond to, respectively, 1-zero shaping and pole-zero shaping with w=800 Hz. An inspection of the noise spectra reveals that the S/Q ratio should be the least for the pole-zero method and the highest for the case without any noise shaping. Notice from Fig. 9 that the pole-zero method redistributes the quantization noise so that it is high at the places the speech spectrum has high amplitudes.

Based on the results of our experiments, we make the following conclusions:

1. Use of noise shaping reduces the perception of the granular noise in the output speech as well as the extent of the limit-cycle problem.

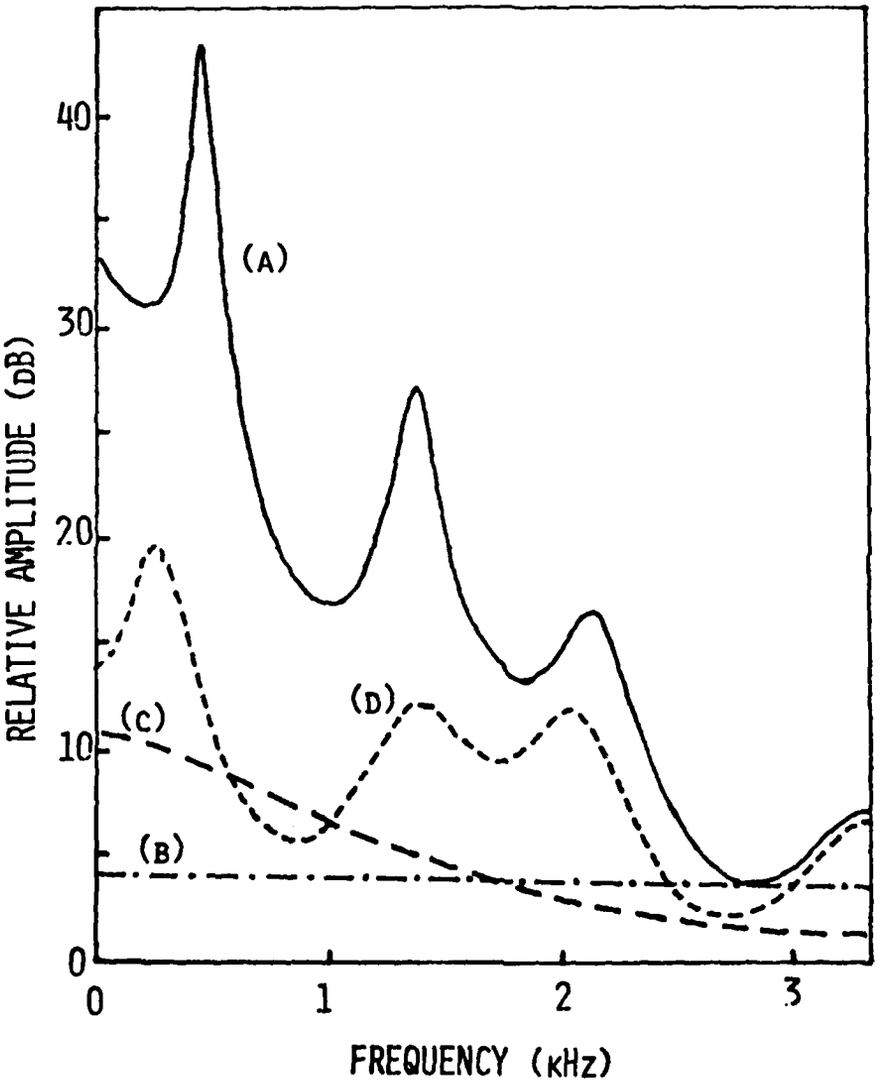


FIG. 9. Comparison of noise shaping methods.

2. The perceptual effect of noise shaping is the most for an APC coder that does not produce any clipping errors; for this case, the pole-zero method produces the best results (see Section 8.4).

3. For APC coders involving clipping errors, the three noise shaping methods produce similar speech-quality improvements, provided no preemphasis is used; the best method for a given configuration may be decided based on the ease of implementation. When preemphasis is used, the pole-zero method produces better speech quality than either of the other two methods.

8. METHODS FOR CODING THE APC RESIDUAL

Recall from Chapter 3 that the output quantization noise is wholly the result of residual coding via the APC feedback loop. Also, the residual constitutes typically about 75-85% of the total bit rate, with the remainder being used for APC parameter transmission. Therefore, proper residual coding is very important. We devoted a substantial part of our effort towards investigating the existing residual coding methods, developing new ones, optimizing them individually, and comparing their relative speech-quality performance. In this chapter, we describe the various coding methods we investigated.

8.1 Chapter Overview

For the discussions given in this chapter, unless stated otherwise, we assume that pitch prediction is not used. For all the coding methods given below, we used the forward-adaptive quantizer described in Section 3.2, with a uniform or an optimal Laplacian nonuniform quantizer.

For the chosen sampling rate of 6.67 kHz, the average number of available bits per residual sample is about 2 bits, with the remaining bits used for the transmission of all other parameters

and their error protection. With only 2 bits available, any method used for coding the residual must have provisions to deal with regions of high amplitude residual samples (e.g., during pitch pulses); otherwise, the resulting clipping errors would significantly degrade the speech quality. Below, we describe five methods of residual coding: Pitch prediction (PP), entropy coding (EC), segmented quantization (SQ), pitch adaptive (PA) coding, and segmented quantization with bit allocation (SQ-BA). Two of these methods (PP and SQ) use a fixed-length code for the residual samples, and the other three use a variable-length code. Given the limited bit resource, each of the five methods attempts to limit the extent of clipping errors in a different manner. The two fixed-length coding methods (PP and SQ) have explicit provisions for reducing clipping errors. The other three methods combat the clipping problem by varying the length of the codeword used for individual samples. Of these three variable-length coding methods, one method (EC) uses a large number of quantizer levels, and the other two (PA and SQ-BA) use only a few possible codeword lengths.

The three variable-length coding methods require variable-to-fixed rate conversion, to be useful in the present fixed-rate transmission application. To avoid involved frame synchronization problems at the receiver [25] and to ensure

reliable decoding in channel error, we chose to accomplish the variable-to-fixed rate conversion over individual frames such that every frame of transmitted data has a fixed number of bits.

Also described below are several composite coding methods obtained by combining two or more of the five basic methods; the composite methods yield significantly better speech quality and higher S/Q ratios. We denote, for ease of reference, the composite methods in terms of the above-introduced abbreviations. For example, the method PP-SQ uses both pitch prediction and segmented quantization.

8.2 Pitch Prediction

8.2.1 The Method

In this method, we use pitch prediction within the APC feedback loop as discussed in Section 3.1, a nonuniform quantizer, and a fixed-length coder for residual samples. The fixed length may be 3 or 4 levels, for example. With 3 levels, we block-code 5 residual samples (a total of 3x3x3x3x3=243 levels) in 8 bits, producing an average of 1.6 bits/sample.

With the use of pitch prediction, we have the choice of employing the S-P or the P-S prediction sequence (see Section

4.1). In our work, we investigated three cases of pitch prediction (1-tap, 3-tap, and 5-tap) and two ways of computing the predictor coefficients c(k): the autocorrelation method and the covariance method [13]. We computed the pitch period M as the nonzero lag corresponding to the peak of the autocorrelation function of the input speech signal in the P-S sequence and of the first residual (see Fig. 3(b)) in the S-P sequence.

Next, we make two observations on the pitch prediction method. First, the pitch-synthesis filter 1/C(z) has a long impulse response (on the order of several pitch periods) because of the large-delay terms (z^{-M}, etc.) of C(z). As a consequence, the speech-quality effect of a channel bit-error or of any other anomaly (see Section 9.3) is propagated for a relatively long duration. However, as will be seen in Chapter 12, the propagation problem in channel error is more-than compensated by the reinsertion of the pitch or harmonic structure into the residual, which would otherwise be significantly distorted.

Second, the use of pitch prediction increases the power gain of the feedback transfer function F(z) in Fig. 5. However, pitch prediction decreases clipping errors and hence increases the W/Q ratio (see Chapter 5); this compensates for the power gain increase and hence limits the net feedback gain.

8.2.2 Stability of Multi-tap Pitch Predictor

The autocorrelation method of computing the predictor parameters guarantees the stability of the filter 1/C(z) for the 1-tap case only, and the covariance method does not guarantee stability at all. The instances of pitch-filter instability were found to cause pops and beeps in the output speech, especially when the quantizer used a small number (e.g., 3) of levels per sample. Our experiments showed that the computed pitch filter was almost always stable when we used the S-P prediction sequence. For the P-S sequence, we investigated two methods of "stabilizing" the multi-tap pitch filter. The 1-tap filter in the covariance method is stabilized by forcing the tap coefficient to be less than 1 in magnitude. These two methods are described below.

8.2.2.1 Switched Prediction

In this method, we check the stability of the pitch filter each frame, and if the filter is unstable, we switch to 1-tap prediction for that frame. Notice that the additional computation involved in computing the 1-tap coefficient is trivial. However, the stability check of the pitch filter involves a significant amount of computation. A straightforward method for the stability check is to obtain the M+m reflection

coefficients of the pitch filter and to check if their magnitudes are all less than 1. The general recursive procedure for computing the reflection coefficients from the taps or prediction coefficients requires a number of multiplies proportional to M^2. However, since there are only 2m+1 nonzero taps, we can reduce the number of multiplies to about 6M.

8.2.2.2 Stable Lattice Prediction

Using the lattice method of linear prediction [26], we have developed a new pitch prediction method. Below, we consider the lattice with only 3 nonzero reflection coefficients K(M-1), K(M), and K(M+1). It is straightforward to derive expressions for these reflection coefficients in terms of the autocorrelations of the speech signal. The lattice method guarantees the stability of the pitch filter. But, the equivalent pitch prediction filter C(z) has five nonzero taps:

$$C(z) = 1+c(1)z^{-1}+c(2)z^{-2}+c(M-1)z^{-(M-1)}+ c(M)z^{-M}+c(M+1)z^{-(M+1)}. \quad (29)$$

The expressions for the five taps in terms of the reflection coefficients are given below:

$$\left.\begin{aligned} c(1) &= K(M)[K(M-1) + K(M+1)], \\ c(2) &= K(M-1)K(M+1), \\ c(M-1) &= K(M-1), \\ c(M) &= K(M)[1 + K(m-1)K(m+1)], \\ c(M+1) &= K(M+1). \end{aligned}\right\} \qquad (30)$$

8.2.3 Experimental Results

For the most part in our work, we experimented with the 1-tap and the 3-tap predictors. We investigated the 5-tap case during the parameter optimization study only (see Chapter 11). Using the P-S prediction sequence and a 4-level nonuniform quantizer, we obtained a S/Q ratio of 15.7 dB for 1-tap pitch prediction and 16.5 dB for 3-tap pitch prediction. The autocorrelation method was used in computing the tap(s) in both cases. Perceptually, the 3-tap case produced more clarity of speech than the 1-tap case.

On the matter of pitch filter stability, we found in one experiment that the 3-tap filter was unstable for about 8% of the frames. The instabilities caused beeps in the output speech when the quantizer used 3 levels per sample. The use of the switched prediction method described above proved to be a successful remedy to the instability problem: The output speech in this case did not contain beeps. Even with a 4-level quantizer, the switched 3-tap prediction method reduced discrete distortions such as pops and clicks. Using again a 4-level quantizer, we

found that the stable 3-coefficient lattice method produced a S/Q ratio of about 15.1 dB, as compared to 15.5 dB in the 3-tap case and 15.7 dB in the 1-tap case reported in the previous paragraph. Listening tests indicated that the lattice method produced "squeaky sounds" and perceivably more quantization noise than the 3-tap case. To understand why the lattice method gave a low S/Q ratio, we examined the values of the 5 taps given by (30). We found that the taps c(1) and c(2) tended to take on values close to ± 1, and that the first part of the filter $(1+c(1)z^{-1}+c(2)z^{-2})$ acted as a preemphasis filter, significantly reducing the spectral dynamic range of the filtered signal. This, in turn, caused the spectral predictor to produce a significantly higher normalized prediction error value [13] than observed in the 3-tap case. This explains the observed drop in the S/Q ratio.

For the P-S prediction sequence, the covariance method of computing the taps produced the same or a slightly lower S/Q ratio than the autocorrelation method. Also, the output speech for the covariance method had low-level "scratchy noises" and clicks. However, using the S-P prediction sequence, we obtained different results. The covariance method produced about 1-2 dB higher S/Q ratios than the autocorrelation method. (We indicate here that with the S-P prediction sequence, we had to use several of the methods given in Chapter 9 to prevent the limit-cycle problem discussed in Chapter 5.)

In conclusion, we make the following observations and recommendations, based on our experimental results:

1. 3-tap prediction produces better speech quality than 1-tap prediction.

2. The pitch filter is almost always stable for the S-P prediction sequence, but this is not so for the P-S sequence. 3-tap switched prediction is a good solution for the instability problem in the latter case.

3. The autocorrelation method is recommended for computing the tap coefficients for the P-S sequence, while the covariance method is recommended for the S-P sequence. (For noisy-channel applications, use of the autocorrelation method is recommended with either prediction sequence; see Chapter 12.)

4. Pitch prediction alone does not provide satisfactory speech quality, as the output speech contains discrete distortions such as pops and clicks.

As will be seen later on in this and several subsequent chapters, pitch prediction, when added to any of the other coding methods, produces better speech quality.

8.3 Segmented Quantization

8.3.1 The Method

The SQ method proposed in [27] employs a nonuniform quantizer and a fixed-length code. The analysis frame is divided into several equal-length segments, and one value of quantizer gain G is computed for each segment. This makes the quantizer

step size adapt rapidly to local variations in the energy of the residual samples. The gain over the whole frame is computed, coded, and transmitted; in addition, the segment gain parameters are computed and their deviations from the (quantized) whole-frame gain (or "delta gain" values) are also transmitted. In our experiments, we used as many as 10 segments.

Ideally, the gain and the delta gain values should be computed for the APC residual, rather than from the first (or the second, with pitch prediction) residual. Since computing the APC residual requires the use of the quantizer, we have a "chicken-or-the-egg" problem. As a suboptimal solution to this problem, the so-called two-spin method is proposed in [27], in which the APC loop is run once with the quantizer gains obtained from the first residual and the resulting APC residual is used in computing an improved estimate for the gains. Except on one occasion (see Section 8.3.3), we used the simple method of computing the gains from the first residual.

8.3.2 Pitch Prediction with Segmented Quantization

The development and extensive investigation of the composite scheme PP-SQ have produced several important results in this work. First, the PP-SQ scheme is computationally the simplest of all the composite schemes we have considered. Second, it is the

only composite scheme that produces a fixed-length code for the residual samples, albeit it offers only two useful codelengths (3 and 4 levels) for a reasonable 16 kb/s coder design. Third, we chose the PP-SQ method for the final design of a robust 16 kb/s APC coder (see Chapter 12).

The SQ method generally provides better results for male speakers than for female speakers, since the dynamic range of the amplitudes of the APC residual within a pitch period is larger for males. The PP-SQ method yields good performance for all speakers, since pitch prediction removes the large peaks of the APC residual at the pitch pulses, and normalization over individual segments rather than over the whole frame provides a better tracking of the short-term amplitude variations of the APC residual.

For the PP-SQ method, the values of gain and delta gains are computed from the second residual E2(z). We found that the delta gains took values over a wide range from about -25 dB to about 5 dB. But, our experiments have shown that satisfactory quantization of the delta gains can be achieved using only 2 bits per delta gain. The nonuniform quantization of the delta gains that we chose from among several other methods is given in Table 4.

Quantizer Input	Level	Quantizer Output
$-\infty$		
	0	-8.0
-5.0		
	1	-3.0
-1.0		
	2	2.0
3.5		
	3	5.5
∞		

TABLE 4. Nonuniform quantization of delta gains

8.3.3 Experimental Results

In one test involving the SQ method with 10 segments and a 4-level nonuniform residual quantizer, we found that the output speech contained several beeps over the six sentences (from the high-quality data base) we processed, because of low values of the W/Q ratio and the resulting excessive quantization-noise feedback. When we used the two-spin method for computing the delta gains, the beeps were replaced by loud scratchy noises. We point out that this excessive noise-feedback problem in this case can be effectively solved by the use of a limiter as discussed in

Section 9.3. Although the beeps were not produced at the output when we used the limiter, the granular noise remained at an objectionably high level. We conclude from this and other tests that the SQ method alone is inadequate.

Before we present the experimental results for the PP-SQ method, we introduce a convenient notation. In this notation, PP3-SQ10, for example, denotes the PP-SQ method with 3-tap pitch prediction and 10-segment segmented quantization. In our initial tests of the PP-SQ method using a 3-level quantizer, we encountered severe problems of limit cycles at the quantizer output. In the next chapter, we present effective ways of preventing the limit cycles. For the rest of this subsection, we consider the use only of a 4-level quantizer.

To investigate the performance effect of varying the number of taps and the number of segments in the PP-SQ method, we tested 6 APC coders obtained by considering two values for the number of taps (1 and 3) and three values for the number of segments (1, 5 and 10). We used a frame size of 25.5 ms and quantized all the parameters except the delta gains. The S/Q ratios computed over six utterances are given in Fig. 10 for these six coders. Relative speech-quality judgments obtained via informal listening are also shown in Fig. 10. Adding segmented quantization (SQ5 or SQ10) to pitch prediction (PP1 and PP3) removed certain "tinkling

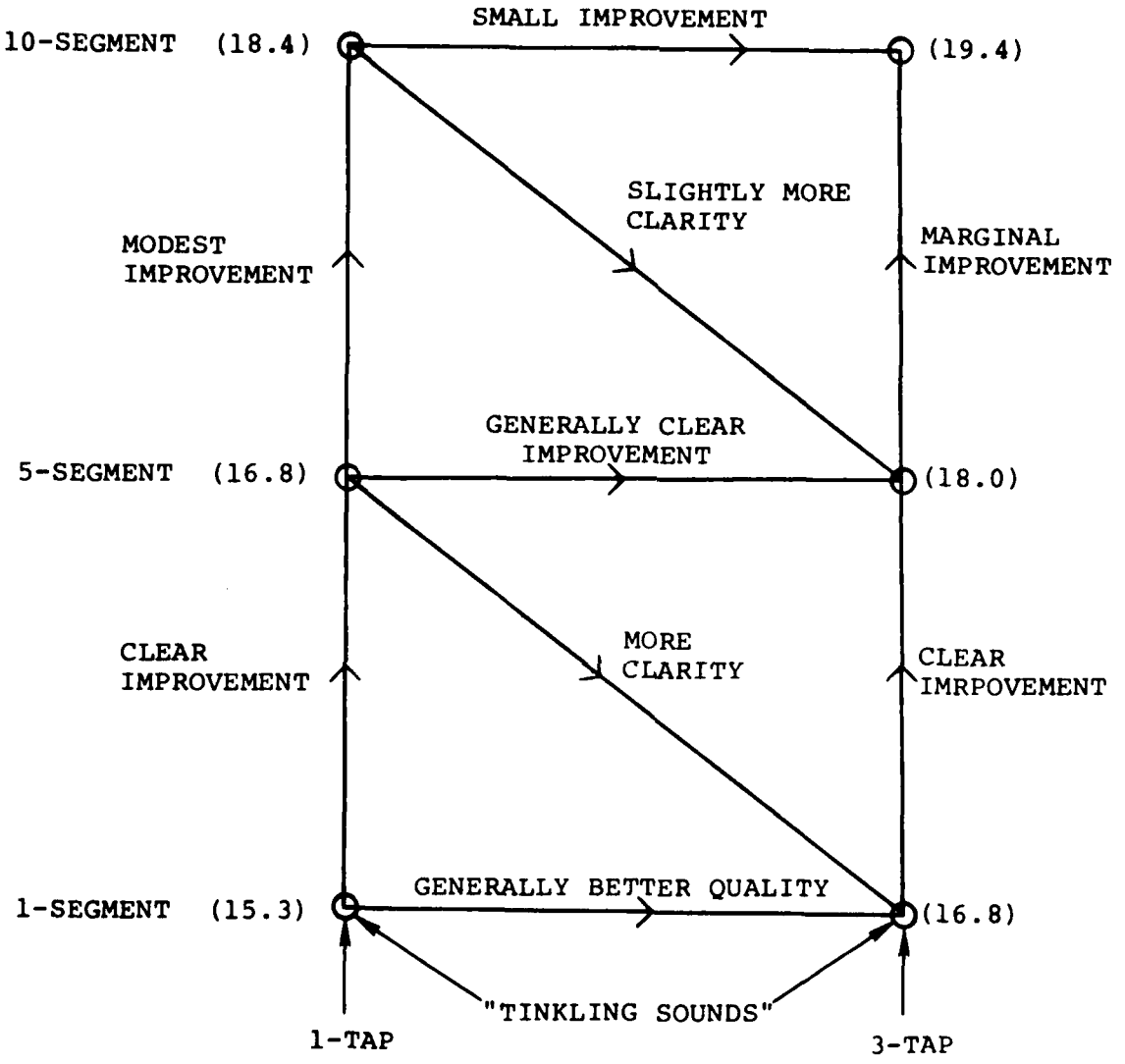


FIG. 10. S/Q ratios and relative speech-quality judgments for 6 PP-SQ APC coders. The number given within parentheses next to each node is the S/Q ratio for the corresponding APC coder. The arrow shown between each pair of APC coders points to the coder judged to produce better speech quality.

sounds" from the output speech and also provided a clear improvement in perceived speech quality. Increasing the number of taps from 1 to 3 generally provided clear improvement only for the 5-segment case. Increasing the number of segments from 5 to 10 yielded a modest improvement for the 1-tap case and only a marginal improvement for the 3-tap case. Of the 6 APC systems, PP3-SQ5 and PP3-SQ10 produced the best overall speech quality.

8.4 Entropy Coding

8.4.1 The Method

This method, described in detail in [5], uses a uniform quantizer with a large (fixed) number of levels to avoid clipping completely. To obtain an average data rate of about 2 bits/sample, the method uses variable-length Huffman or entropy coding [28]. In entropy coding of the residual, frequently occurring residual values (those close to zero) are coded with a small number of bits, and infrequently occurring values (large amplitudes, usually in the clipping range of fixed-length quantizers) are coded with a large number of bits in such a way that the average bit rate is minimized. Following reference [5], we used a (suboptimal) self-synchronizing entropy code, with codewords given by: 0, 10, 110, 1110, etc., because of our noisy-

channel application. Notice that in case of a channel bit-error, the decoding error is in the form of either merging two residual samples or splitting one into two.

The fixed step size of the uniform quantizer is a parameter, whose value is experimentally computed to obtain the desired average number of bits/sample. Increasing the step size decreases the entropy of the code by forcing more samples into the level coded with one bit; this decreases the required average number of bits per sample. Similarly, decreasing the step size increases the average number of bits per sample. With the adaptive quantizer in Fig. 2, changing the quantizer step size can be accomplished by changing the quantizer gain G. To interface the variable-rate coder to a fixed-rate channel, we require variable-to-fixed rate conversion, which is described next.

8.4.2 Variable-to-Fixed Rate Conversion

The scheme we used for achieving this conversion was developed as part of another government contract at BBN [29]. In applying this scheme, we made some modifications to improve its performance. Described below is an outline of the rate conversion scheme.

The scheme forces the number of bits to be a constant at each frame by changing the gain factor in front of the quantizer and repeating the analysis of the APC loop for the whole frame. The new residual thus obtained is coded and the number of bits used is computed. The process is repeated iteratively until the desired convergence is achieved. As a practical matter, we limited the number of iterations to 5. At the end of the 5th iteration, the method chooses the gain value that yielded a number of bits that is closest to, but not exceeding, the desired number; the difference is made up by inserting one-bit codes or "filler bits." The analysis of the APC loop is then repeated with this final gain value; this step avoids having to store all the intermediate residual codewords obtained in the five iterations. In case all five iterations provided total bits for the frame above the desired value (which we encountered a few times in our experiments), the method chooses the one that came closest to the desired number and discards as many of the residual samples from the end of the frame as required. The discarded samples are assumed to be zero at the receiver.

The gain adjustment mentioned above is performed using a procedure outlined next. Denote by d_B the difference between the actual number of bits provided in an iteration and the desired number. During the initial iterations, until a "zero-crossing"

is obtained in the sense that δB changes sign, the method adjusts the gain using a modified version of the so-called 6 dB/bit rule, i.e., the gain adjustment in dB is given by $(6+|\delta B|/20)\delta B/N$, where N is the number of samples in the frame and the term $|\delta B|/20$ is the modification we found to yield good convergence. Once the "zero-crossing" is obtained, the method adjusts the gain using the modified false position method of computing the zero of a function [30].

Since the number of bits used per frame is quite sensitive to even small changes in the gain G (e.g., 0.1 dB), we use the unquantized G at the input of the quantizer and the quantized G at its output (see Fig. 2); to minimize the mismatch between the two gain values, we use 10 bits for quantizing G.

8.4.3 Experimental Results

In testing the variable-to-fixed rate conversion scheme, we found that the number of levels used by the quantizer had to be increased from the previously used value of 19 to 43, to produce about the same perceived speech quality as from the free-running variable-rate system. The S/Q ratio, however, dropped about 1 dB, because of the gain or step size adjustment required by the rate conversion scheme. The transmission rate of the filler bits was found to be about 5 bits per frame or about 200 b/s.

We combined pitch prediction with entropy coding to produce the composite scheme EC-PP. Our investigation has clearly shown that for the same bit rate, EC-PP1 produces better speech quality than EC, and EC-PP3 produces further improvement over EC-PP1. In one test, we used a frame size of 25.5 ms and 1-zero noise shaping. The three 16 kb/s coders, EC, EC-PP1, and EC-PP3, yielded the S/Q ratio values: 19.9 dB, 20.8 dB and 21.2 dB.

Use of noise shaping in the entropy coding method produced significantly better speech quality in the form of reduced granular noise relative to the case without noise shaping. We report an interesting experiment that demonstrates the importance of the type of noise shaping used in the EC method. In the initial stages of this work, we mostly used the 1-zero noise shaping method. With this noise shaping, the 4-level PP1-SQ10 method with a S/Q ratio of 18.2 dB produced a perceivable speech quality improvement (in particular, reduced quantization noise) over the EC-PP3 method with a S/Q ratio of 21.2 dB, notwithstanding the S/Q ratio difference of 3 dB. However, when we later used the pole-zero noise shaping method we observed a reversal in the speech quality ordering of the two coders. The EE-PP3 system with pole-zero noise shaping (w=800 Hz) yielded a reduced S/Q ratio of about 19.0 dB, but produced speech that was significantly better than from the same system with 1-zero noise shaping.

8.5 Pitch-Adaptive Coding

The original idea (see Section 8.5.1) that led to our development of the PA method described below was recently proposed in [31], as a way of efficiently allocating the available bit-rate resource for time-domain residual quantization. The basic PA method we describe in Section 8.5.2 is far simpler in both side-information transmission requirements and computational complexity than the method proposed in [31]. Also included in Section 8.5.2 is a simple method of obtaining a fixed number of bits/frame. We have made several modifications to the basic PA method, producing a viable and effective pitch-adaptive coding method that includes pitch prediction and pitch-synchronous segmented quantization (rather than the time-synchronous SQ discussed above in Section 8.3). This novel PA-PP-SQ coding method represents a significant contribution of this work. This notation, although long, allows us to specify conveniently the number of segments in a pitch period, the number of pitch taps, and the number of transmitted delta gains (e.g., PA4-PP3-SQ3). Since the development of the PA-PP-SQ method was based on the experimental results from our earlier versions of the PA method, we present below the experimental results as part of the initial subsections (unlike in the previous Sections 8.2-8.4).

8.5.1 Itakura's Method

This method divides each pitch period of the first (LPC) residual into a small number, e.g., 4, of equal-length segments. The energy per sample E_i is computed for each segment. Let B_i be the number of bits used to quantize the residual samples in the ith segment, where i = 1,2,3,4. For convenience, the indexing is chosen such that $E_1 \geq E_2 \geq E_3 \geq E_4$. Let B_0 be the average of the four B_i values. If the pitch period is M samples, the problem is then to allocate the MB_0 bits so that the mean-square quantization error is minimized. It can be shown [6] that for the optimal case, the mean-square errors in individual segments should be equal, and B_i is given by

$$B_i = B_0 + 1/2 \log_2[E_i/(\prod_{k=1}^{4} E_k)^{1/4}]. \qquad (31)$$

This means that the segment containing the pitch pulses (segment numbered 1) is assigned the maximum number of bits, and the segment numbered 4 is usually assigned the fewest bits.

The side information transmitted to the receiver consists of the following quantities: pitch period value(s) and as many sets of E_i and of locations of segments as there are pitch periods in the frame. The method requires the calculation of one bit assignment per pitch period at both the transmitter and the

receiver. The details of locating pitch periods and locating segments within a pitch period are not given in [31]. Also, the reference does not consider the issue of fixed-rate transmission.

We make two remarks on Itakura's method. The first remark deals with the apparent similarity between this method and ATC. There is an important difference between the two methods: ATC operates in the frequency (transform) domain, while the above method functions in the time domain. In ATC, some formant peaks may fade in and out of a frequency band, which causes time-varying effects usually perceived as clicks. Such fade-in and fade-out events can also occur in the above method, but they happen in the time domain and thus may not produce a perceptually degrading effect. Second, although Itakura's method computes pitch, it does not use pitch prediction.

8.5.2 Basic Pitch-Adaptive Method

The basic PA method employs a forward-adaptive nonuniform quantizer that uses a variable number of bits/sample. Unlike the EC method, this method uses only a small number (e.g., 4) of code-lengths for the residual samples.

Figure 11 illustrates the basic PA method by means of a 4-segment example. As shown in the figure, one value M of the

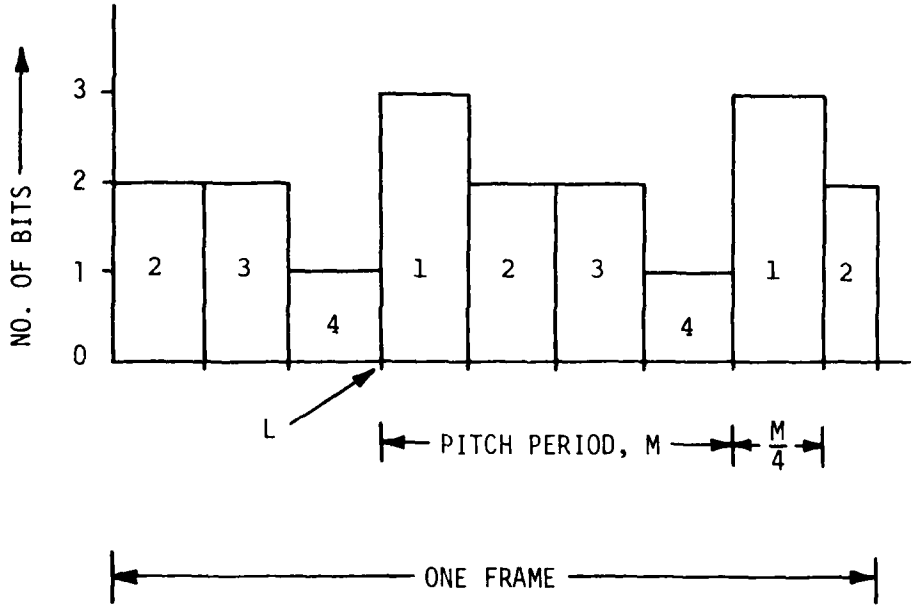


FIG. 11. Bit assignment in the basic pitch-adaptive coding scheme.

pitch period is used in dividing the whole frame into segments of M/4 samples each. The segment at the end of the frame has fewer than M/4 samples. We explain later how to accommodate cases where M/4 is not an integer. For the example in the figure, segments numbered 1 are quantized using 3 bits, segments numbered 2 and 3 are quantized using 2 bits, and segments numbered 4 are quantized using 1 bit. The average number of bits per sample is approximately 2. This allocation of bits among the segments, denoted by {3,2,2,1} for this example, is fixed in time, so that this information is not transmitted to the receiver. The side information to be transmitted to the receiver consists of three quantities: the pitch period M, the location L of the beginning of the very first segment numbered 1, and a 1-bit code to be defined in the next subsection. Notice that L can take only 4 values: 0, M/4, M/2, and 3M/4. For the frame shown in the figure, L=3M/4. For the 4-segment example, L is transmitted using 2 bits.

Let us consider the case when M/4 is not an integer. If (M-J)/4 is an integer, where J can be 1, 2, or 3, then 3 of the 4 segments over each pitch period are chosen with (M-J)/4 samples. The segment with the assigned bits per sample closest to the desired average bits per sample is made to contain J+(M-J)/4 samples.

The above discussion has considered only voiced frames. The case of unvoiced frames is treated as part of the next subsection dealing with the procedure used to compute L.

8.5.2.1 Computation of the Location Parameter

Using optimal nonuniform unit-variance Laplacian quantization, we have precomputed and stored in memory the quantization tables corresponding to the various numbers of bits/sample and the corresponding mean-square quantization errors. For a given frame, we determine the optimal value of the location parameter L as follows. For each allowable value of L, which uniquely defines the segmentation procedure as discussed above, we compute the average over different segments of the quantity which is the product of the sum of the squares of the residual samples in a segment and the stored mean-square quantization error for that segment. This average error measure is computed for two cases: nonuniform bit allocation among segments (e.g., {3,2,2,1}), and uniform bit allocation (e.g., {2,2,2,2}). The two cases are coded using a 1-bit code U/NU (uniform/nonuniform). The values of L and U/NU that yield the least average quantization error are used in the PA scheme for that frame. The inclusion of the uniform bit allocation case allows handling of the unvoiced frames as well. The example

uniform bit allocation given above, namely {2,2,2,2}, is clearly for the case when the desired average over the frame is 2 bits/sample. If this average is not an integer, say, 1.75 bits/sample, then a bit allocation such as {2,2,1,2} may be used for the "uniform" option. In this last case, choosing an optimal value of L is meaningful even for unvoiced frames.

Two remarks are in order. First, since L can take only a small set of values (e.g., 4), an exhaustive minimization over this set to compute the optimum L is quite reasonable. Second, computation of segment energies for each L can be simplified by computing once and storing the squares of all the samples over the frame. (*Segment energies by themselves cannot be stored* since segment widths and locations change with L.)

8.5.2.2 Variable-to-Fixed Rate Conversion

This procedure yields a fixed, prespecified number of bits over each frame. Let B_0 denote the average number of bits per sample corresponding to this desired total number of bits. For a given value of L and the associated segment assignment, the "ideal" periodic bit allocation is assigned for the same, and the total number of bits used for the frame is computed. If this total exceeds the desired number, the following action is taken. Starting from the beginning of the frame, a search is made for a

segment boundary where a segment with its assigned bits/sample less than the average B_0 precedes a segment with its assigned bits/sample equal to or greater than B_0. This segment boundary is then shifted to the right (more precisely, towards the end of the frame), to increase the size of the first segment by one sample. If the resulting total is still too high, the segment boundary is shifted again at a place one pitch period later. At the end of the frame, if the total is still too high, the above process is repeated. On the other hand, when the ideal bit allocation mentioned at the beginning of this paragraph leads to a total number of bits that is less than the desired total, then the shifting of segment boundaries is done in the opposite direction.

The above iterative process typically converges within a few iterations. Unlike the procedure used in the entropy coding method (see Section 8.4.2), this procedure does not actually quantize the residual samples until the proper segment and bit assignment have been decided; also, the individual iterations in this method are relatively simple to perform.

8.5.2.3 Comparison with Segmented Quantization

For the PA method described above, each segment is made up of similarly located samples from the successive pitch periods of

a frame. In this sense, the PA method employs pitch-synchronous segments. In contradistinction to this, the SQ method uses time-synchronous segments. More important, on the one hand, the SQ method, by using different quantizer gains over individual segments, tracks the residual amplitudes by expanding or contracting the quantizer step size. On the other hand, the PA method uses the same gain and hence the same step size over the whole frame, but tracks the residual amplitudes by increasing or decreasing the number of bits. One can show that the PA method gives a higher S/Q ratio than does SQ. The above interpretation of pitch-synchronous segments was used in developing the PA-PP-SQ scheme described in Section 8.5.3.

8.5.2.4 Experimental Results

In our experimental investigation of the basic PA method, we found that the output speech contained occasional beeps. When we added pitch prediction to the PA method, this limit-cycle problem was eliminated. However, in our tests, a 4-segment, 1-tap PA4-PP1 method produced a S/Q ratio of only 17.1 dB. One reason for this may be that the residual samples within a segment are quantized using a unit-variance quantizer, but they do not have unit variance. A solution to this problem, discussed in the next section, is to employ segmented quantization, but using the pitch synchronous segments just defined above in Section 8.5.2.3.

8.5.3 Pitch Prediction and Pitch-Synchronous Segmented Quantization

In this method, the quantizer of the APC residual uses for each pitch-synchronous segment: (1) a different gain G, which is computed from the second residual for the same pitch-synchronous segment; and (2) in general, a different number of bits/sample. For those frames for which the uniform bit allocation is chosen (see Section 8.5.2.1), the time-synchronous segmented quantization method is used.

The resulting pitch-adaptive method, denoted by PA-PP-SQ, was found to provide a significant increase in perceived speech quality and S/Q ratio over the PA-PP method. For the 1-tap, 4-segment case with the fixed bit allocation {3,2,1,2}, the PA4-PP1-SQ4 method produced a S/Q ratio of 19 dB, which is 1.9 dB higher than the S/Q ratio of the PA4-PP1 method. As another interesting comparison, the 1-tap pitch prediction alone produced a S/Q ratio of only 15.3 dB.

We performed several experiments to investigate various aspects of the PA-PP-SQ method. The results of all but two experiments are stated below briefly, followed by a discussion of the other two experiments.

1. Transmission of M/4 (for the 4-segment case) instead of M, to save bits required for pitch transmission, produced perceivable speech-quality distortions.

2. Transmission of the location parameter L using 3 bits instead of 2 bits, for the 4-segment case, did not yield any perceivable improvement in the speech quality. (Note that by increasing the accuracy of L, the segmentation procedure allows the frame to begin with a less-than-full segment.)

3. For an average of about 2 bits/sample, out of the several 3-, 4-, and 5-segment cases we tested, we found that the 4-segment case with the nonuniform bit allocation {3,2,1,2} produced the highest S/Q ratio.

4. No perceivable speech-quality degradation resulted when we combined, for the purpose of segmented quantization, segments within a pitch period with the same number of bits per sample. This means, for the example with the bit allocation {3,2,1,2}, only 3 delta gains need be transmitted. We denote this case explicitly with the notation PA4-PP1-SQ3, for example.

5. Delta gains can be coded using 1 bit for the 1-bit and 2-bit segments, and using 2 bits for segments with larger number of bits/sample.

Comparison with Time-Synchronous Segmented Quantization: In this experiment, we used time-synchronous segmented quantization for all frames rather than only for frames using uniform bit allocation. The segments that this method considers for amplitude normalization are different from those that the above PA method considers, for nonuniform bit allocation. The results of our experiments showed that for the 4-segment case, pitch-synchronous SQ was better than time-synchronous SQ. The former method produced, as noted above, an S/Q ratio of 19 dB. To produce the same value, the number of segments for the time-synchronous method had to be increased to 10.

Adaptive Bit Allocation: In this experiment, we used an adaptive, rather than a fixed, bit allocation, to track frame-by-frame variations in the residual amplitude envelope. A two-bit code was transmitted to indicate which of the following four bit-allocations the current frame employed: {3,2,2,1}, {3,2,1,2}, {3,1,2,2}, and {2,2,2,2}. The adaptive scheme produced only a slight improvement in perceived speech quality (and about the same S/Q ratio) over the fixed case. We feel that the added complexity is not justified by the small improvement.

8.6 Segmented Quantization with Bit Allocation

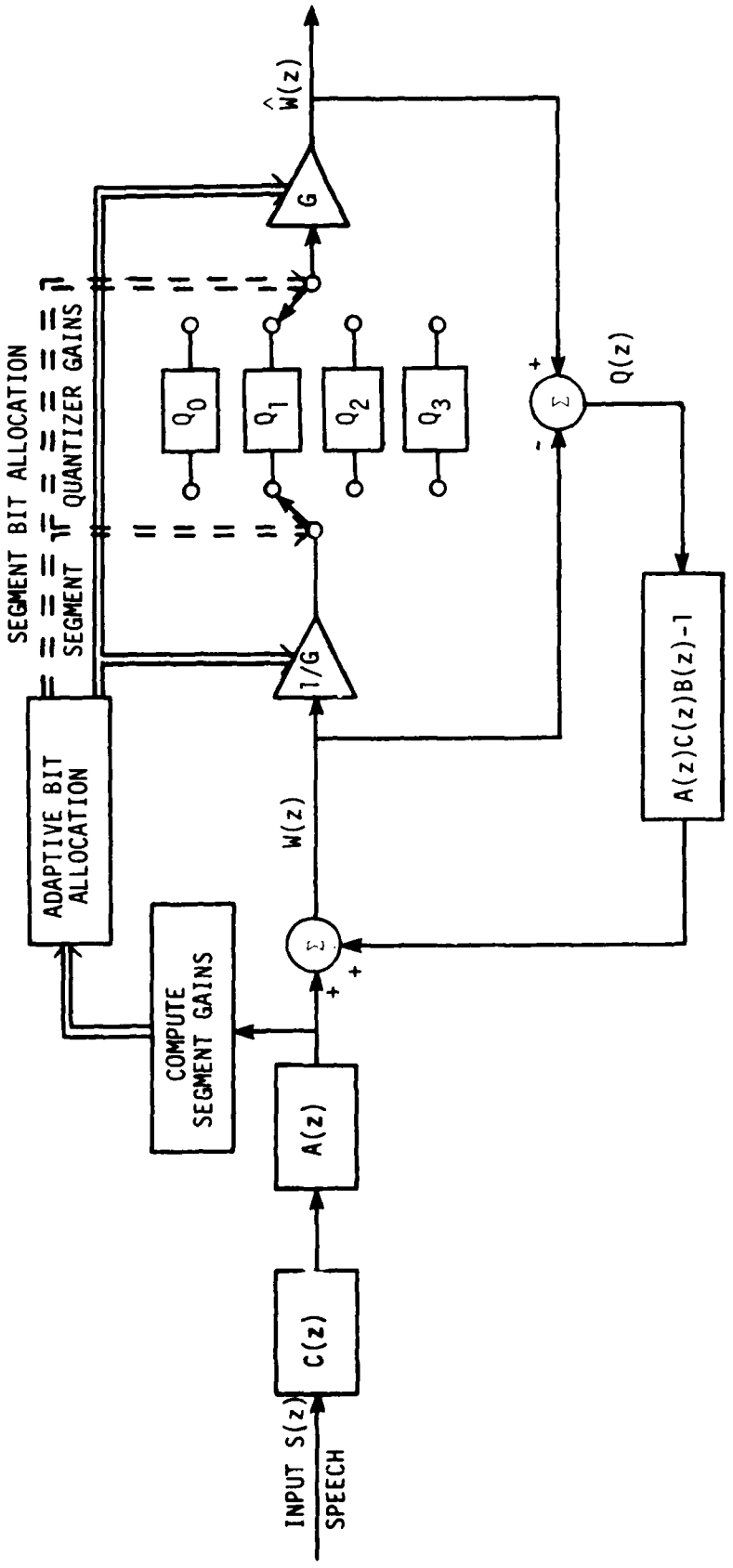
8.6.1 The Method

The SQ-BA method employs a nonuniform quantizer that uses a variable number of bits/sample. This method divides each frame into a fixed number, N, of equal-length segments. The samples in segment i are quantized using B_i (an integer) bits. The bit allocation to be used $\{B_i, 1 \leq i \leq N\}$ is coded and transmitted each frame as side information. A set of optimal n nonuniform unit-variance (Laplacian) quantization tables are stored in memory, where n is the number of distinct values of B_i. In the coding and decoding of the residual samples in the i-th segment, the quantizer gain corresponding to that segment and the quantization

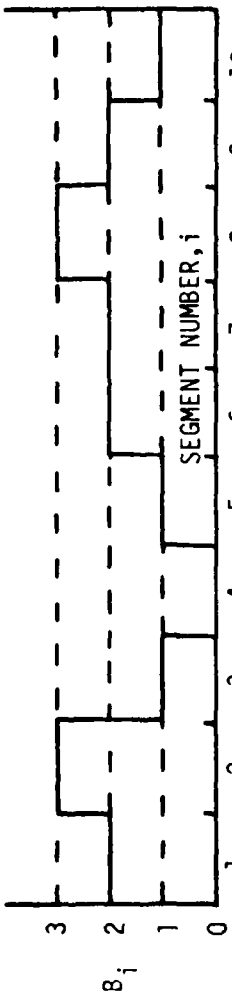
table corresponding to B_i are used. For the case $B_i=0$, all samples in that segment are decoded as zero. Thus, we have combined adaptive bit allocation with segmented quantization, using the same segments for both operations. As discussed below, the quantizer gain of each segment can be obtained from the bit allocation $\{B_i\}$ and the overall gain of the frame. Therefore, a single set of codewords is used to transmit both the bit allocation and the segment gains.

Figure 12(a) depicts an APC-NF system using the SQ-BA method (with pitch prediction). In the figure, Q_i indicates an i-bit quantizer, and the segment bit allocation indicated by the dashed double lines controls the choice of the quantizer via the switch arrangement shown. Figure 12(b) shows the bit allocation for the 10 segments for a frame.

The bit allocation in each frame is chosen to minimize the mean-square quantization error under the constraint that the total number of bits per frame be equal to a given value. The method we have used for determining such an optimal bit allocation is similar to the one used in ATC [6]. Briefly, the optimal allocation is $B_i=B_0+(P_i-P)/S$, where B_0 is the average number of bits per sample, P_i is the gain G_i in dB of segment i, P is the geometric mean of the segment gains, (i.e., the average of P_i, $1\leq i\leq N$), and S is a constant equal to 6 (dB/bit) for fine



(a) Block diagram of the APC-NF system using the SQ-BA coding scheme.



(b) Example bit allocation for a frame, for the 10-segment case.

FIG. 12. Segmented quantization with bit allocation.

uniform quantization. For coarse or nonuniform quantizers, the value of S must be chosen experimentally. The set of B_i's must be rounded to integers and still satisfy the constraint on the total number of bits per frame. To do this, a simple iterative algorithm is used, which typically requires only a small number of iterations. A step-by-step description of the bit-allocation algorithm is given in Fig. 13. Notice that Steps 9 and 10 in this figure give the expressions for the segment gains to be used in the amplitude normalization of the segment residual samples. Since the bit allocation $\{B_i, 1 \leq i \leq N\}$ and the quantized frame gain $\hat{P}$ are transmitted to the receiver, the segment gains $\hat{G}_i$ are also computed at the receiver from the same expressions.

Below, we make several remarks comparing the SQ-BA method just described and the other coding methods. First, the SQ-BA method is different from the SQ method in at least two ways: (1) a variable number of bits/segment rather than a fixed number and (2) transmission of the segment gains via the bit allocation rather than via the delta gains.

Second, although the SQ-BA method uses a bit-allocation procedure similar to the one used in ATC, there are significant differences between the two approaches. The question of time-domain versus frequency-domain coding has already been discussed in Section 8.5.1. With the algorithm used in ATC, the segment

1. $P_i = 20 \log_{10}(G_i)$, $1 \leq i \leq N$.

2. $P = \frac{1}{N} \sum_{k=1}^{N} P_k$

   $\Delta B = 0$

3. $P = P + \Delta B.S$

4. $\hat{B}_i = B_0 + (P_i - P)/S$

5. $\hat{B}_i = [B_i]$ (truncation)

6. $\Delta B = \frac{1}{N} \sum_{k=1}^{N} \hat{B}_i - B_0$

7. If $|\Delta B| > \varepsilon$ (tolerance), go to Step 3

8. Quantize P to $\hat{P}$

9. $\hat{P}_i = \hat{P} + S(\hat{B}_i - B_0)$

10. $\hat{G}_i = 10^{\hat{P}_i/20}$

FIG. 13. A step-by-step description of the bit-allocation algorithm used in the SQ-BA method.

gains would be coded and transmitted directly; the bit allocation would then be determined from the set of decoded segment gains by means of the algorithm, which would be used in both the transmitter and the receiver. In the SQ-BA method described above, the bit allocation is computed only at the transmitter from the unquantized segment gains and explicitly transmitted to the receiver. Segment gains to be used in both the transmitter and the receiver are computed from the bit allocation as explained above. The difference between these two approaches is apparent in the presence of channel bit-errors. In the ATC approach, a single bit-error on a segment gain may cause errors in any or all of the B_i's computed at the receiver. Thus, the single bit-error can lead to erroneous decoding of all the samples in that frame. In the approach we have used, a single bit-error causes erroneous decoding of only the samples in the corresponding segment and in the segments that follow.

Finally, we have developed the SQ-BA method (1) as an alternative to the entropy coding method in terms of producing both less computational complexity and potentially better channel-error performance and (2) as an alternative to the pitch-adaptive method in terms of providing an easier implementation on the MAP-300 array processor. The data-dependent segment sizes and locations, among other things, make the PA method extremely difficult to implement on the MAP.

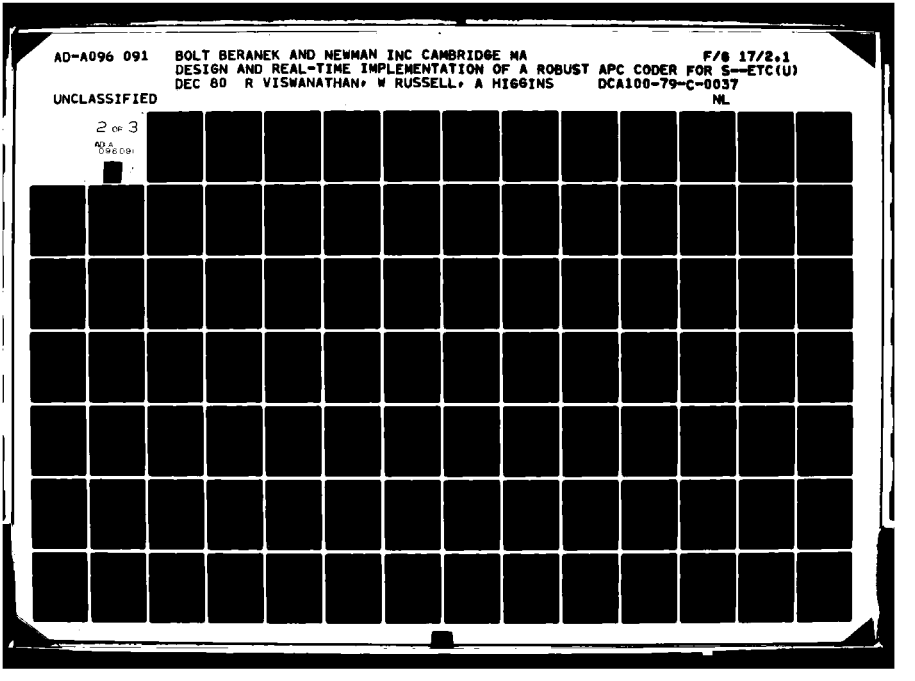
8.6.2 Experimental Results

The SQ-BA method without pitch prediction produced noticeable roughness in the output speech; the roughness was eliminated when we added pitch prediction. The effect of the number of pitch filter taps for the PP-SQ-BA system on speech quality was found to be the same as for the PP-SQ system. Under the 16 kb/s constraint, 3-tap pitch prediction produced better speech quality than 1-tap prediction. We briefly experimented with the choice of the possible values for the number of bits/sample B_i. Considering 4 values, we tested the two sets: $\{0,1,2,3\}$ and $\{1,2,3,4\}$. The first set produced significantly higher S/Q ratios than the second set, under all conditions we tested. We also investigated the use of only 2 values $\{1,2\}$ for B_i. This case produced lower S/Q ratios than the above two cases.

If 10 segments are used, the average number of bits/sample B_0 can be any multiple of 0.1 (e.g., 1.8, 1.9, etc). This flexibility gives a wide choice of systems for investigating the tradeoff involving frame size, number of taps, and B_0 (see Section 10.5 for more results). In one test, we employed 5-tap pitch prediction and found that at 16 kb/s it produced similar quality to that from the 3-tap case. In another test, we observed that using a frame size of 27 ms or larger caused

AD-A096 091 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2.1
DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR S--ETC(U)
DEC 80 R VISWANATHAN, W RUSSELL, A HIGGINS DCA100-79-C-0037
UNCLASSIFIED NL

2 OF 3
AD A
096 091



"clicks" and "dropouts" in the perceived speech. These degradations were mitigated but not eliminated by increasing the number of segments per frame.

We experimented with different types of nonuniform quantizers. We found that using a Gaussian quantizer produced speech of similar quality but a slightly lower S/Q ratio than using a Laplacian quantizer. In another experiment, we used a Gaussian 1-bit quantizer, a Laplacian 2-bit quantizer, and a gamma 3-bit quantizer. This resulted in lower speech quality but a slightly higher S/Q ratio, as compared to the case of all Laplacian quantizers. We used the original scheme using Laplacian quantizers in our subsequent work. For Laplacian quantizers, the value S=4.0 dB/bit was found to give the best results. (The parameter S is used in Steps 4 and 9 shown in Fig. 13.)

In all our tests of the PP-SQ-BA method, we found that it produced S/Q ratios approximately equal to or less than those from the PP-SQ method under similar conditions, although the speech quality produced by the former method was similar to or sometimes better than that given by the latter method. While we have not thoroughly investigated this issue of unexpectedly low S/Q ratios, we feel that a possible explanation for its cause is that the bit allocation computed from the second residual does

not adequately match the APC residual, since the two residuals can have different segmental amplitude variations.

9. LIMIT-CYCLE BEHAVIOR OF THE APC SYSTEM

9.1 Discussion of the Problem

As reported in Chapter 5, when the feedback gain G_F of the APC loop is positive, the quantization noise Q(z) builds up to excesive values, causing the quantizer output $\hat{W}(z)$ to exhibit a limit-cycle behavior. Depending on the severity and duration of this behavior, the coder output is perceived as the non-speech-like sounds beeps and glitches or as speech containing discrete distortions e.g., clicks, pops, etc. We noted in Chapter 5 that positive values of G_F are caused by excessive values of the power gain of the filter F1 = ABC-1, inadequate quantization accuracy (which results in low values of the W/Q ratio), or both. In the last chapter, we reported that the basic versions of several of the residual coding methods provide inadequate quantization accuracy and hence cause varying extents of the limit-cycle problem. The various composite coding methods, on the other hand, were found to nearly eliminate the limit-cycle problem, provided the coder uses (1) the P-S prediction sequence and (2) an average number of bits/sample close to 2. Considering the issue of power gain, we reported in Chapter 7 that noise shaping helps to reduce the power gain. In fact, it was found that increasing the amount of pole-zero noise shaping by increasing

the bandwidth parameter w from 200 Hz to 800 Hz eliminated the limit-cycle problem in an entropy-coding APC system, in which, because of its high W/Q ratio, the limit-cycle problem occurs only very infrequently. For all other coding methods we investigated, noise shaping by itself does not provide sufficient reduction in the power gain. In this chapter, we discuss two other methods for reducing the power gain: high-frequency correction (Section 9.4) and preemphasis (Section 9.5).

The limit-cycle behavior can also be triggered if the pitch prediction filter used within the APC loop is unstable. (The spectral filter is always stable since we use the autocorrelation method of linear prediction.) As we reported in Section 8.2, the switched prediction method provides pitch filter stability without perceivably degrading the coder performance.

Another approach towards solving the limit-cycle problem, discussed below in Section 9.3, is to limit or clamp the filtered quantization noise at some value when it is building up. Before we discuss this approach and others mentioned above, we present in the next section our experimental observations regarding the extent of the limit-cycle problem for the two prediction sequences. Throughout our experimental investigation of the limit-cycle behavior of APC, we used several test cases to examine if and how a given approach to cure the problem

controlled the extent of the problem. Some of these test cases are mentioned below as part of our presentation of the experimental results.

9.2 S-P versus P-S Prediction Sequence

At the beginning of this project, we used only the S-P prediction sequence. While the entropy-coding method with a large number of quantizer levels produced no limit-cycle problems, the fixed-length coders that we subsequently implemented caused severe distortions to be perceived in the output speech. For example, the PP1 system using 2 bits/sample and 25.5 ms frame size produced loud beeps in the output speech at the rate of one or two per sentence, with each beep lasting about 50 ms. Even with 4 bits/sample, we encountered one beep over 6 sentences of a total duration of about 15 sec. Limit-cycle problems were also encountered with the PP-SQ system.

When we switched to the P-S prediction sequence, we found that for the case of 2 bits/sample, all non-speech-like sounds were eliminated, and the discrete distortions were significantly reduced. The primary reason for this improvement is that using the P-S sequence produced power gain values that were about 1-2 dB lower than those obtained using the S-P sequence. However,

the PP3-SQ10 system with the P-S sequence, a 3-level quantizer, and a frame size of 18 ms still produced beeps at the output.

In summary, the S-P prediction sequence is more likely to encounter the limit-cycle problem than the P-S sequence, since the former yields higher power gain values.

9.3 Saturating Limiter

In this method, the magnitude of the filtered quantization noise is limited to some reasonable value [11]. The APC-NF configuration shown in Fig. 5 serves best to explain the idea of this method and to suggest a way of computing the value of the limit. Since the APC residual W(z) is given by Eq. (9), by limiting the magnitude of the filtered noise F(z), we can ensure that W(z) is not dominated by the quantization noise. Eq. (9) also suggests that the filtered noise samples f(n) may be limited in magnitude to θ times the rms value of the second residual E2(z). Notice that this rms value is already computed for setting the quantizer gain. As in [11], we used a value of θ=2. We interpret the saturating limiter as serving the role of a "safety valve."

We implemented the limiter first in the APC-NF configuration and investigated its effectiveness in several test cases. The

first case we considered is an SQ10 coder (with no pitch prediction) using 2 bits/sample and 25.5 ms frame size. Without the limiter, this coder produced about 1-2 beeps per sentence. Use of the limiter in this coder was found to entirely eliminate all beeps and other discrete distortions. The S/Q ratio dropped slightly from 16 dB (no limiter used) to 15.8 dB (limiter used). Another interesting case we tested is a PP3-SQ4 coder using the P-S sequence, with 3-level quantization and 19.5 ms frame size. In this case, we found that the limiter was activated for about 0.3% of the filtered noise samples. The S/Q ratio dropped from 14.9 dB to 13.2 dB because of the limiter. Using the limiter in this case eliminated all the beeps and other discrete distortions, but some of the processed sentences sounded objectionably reverberant. The reverberant quality is due to the periodic propagation by the filter $1/C(z)$ of the clipping "errors" introduced by the limiter. This fact can be easily shown as follows. The limiter's output can be written as

$$F'(z) = F(z) + D(z), \tag{32}$$

where $D(z)$ is the limiter-caused clipping noise. From Fig. 5, we obtain

$$\begin{aligned} W(z) &= E2(z) + F'(z) \\ &= A(z)C(z)S(z) + [A(z)C(z)B(z)-1]Q(z) + D(z). \end{aligned} \tag{33}$$

From the receiver shown in Fig. 1, the output speech R(z) is given by

$$R(z) = [1/C(z)][1/A(z)][W(z)+Q(z)]. \quad (34)$$

Using (33) in (34), we obtain

$$R(z) = S(z) + B(z)Q(z) + [1/C(z)][1/A(z)]D(z). \quad (35)$$

Thus, the reconstructed speech has a component that is the clipping noise D(z) filtered by 1/A(z) and 1/C(z). This explains the cause of the reverberant quality mentioned above. It can be easily seen that the only way to avoid this problem is to avoid placing the limiter in the path of the pitch predictor within the APC loop. Such a limiter placement is possible for the APC-PF system with either of the two prediction sequences and for the APC-HF system with the S-P sequence. The output speech expressions in the two cases are given below:

$$R(z) = S(z) + B(z)Q(z) + D(z), \quad \text{(APC-PF)}, \quad (36)$$

and

$$R(z) = S(z) + B(z)Q(z) + [1/A(z)]D(z), \quad \text{(APC-HF)}. \quad (37)$$

Notice from (37) that the clipping noise is shaped by the

spectral filter for the APC-HF case. Our experiments showed that the limiter used in the APC-PF configuration did not produce any appreciable effect in terms of alleviating the limit-cycle problem. As expected, using the limiter in the APC-HF system did not produce any reverberant quality, and it reduced the severity of (and in some cases eliminated) the limit-cycle problem.

9.4 High-Frequency Correction

This technique, proposed in [11], is a way of reducing the power gain of the predictor A(z), $G_p(A)$. Notice that $G_p(A)$ is simply the integral of the power spectrum of A(z). This inverse spectrum has large amplitudes at the high frequencies near the cutoff frequency of the anti-aliasing (A/D) lowpass filter; the large amplitudes are primarily due to the nonideal lowpass filters. Reference [11] suggests a simple and effective method of reducing $G_p(A)$. In this method, the high-frequency amplitudes of the power spectrum of the signal used for LPC analysis are increased ("corrected") by digitally adding to that signal a highpass-filtered white noise. Since we use the autocorrelation method of LPC analysis, the high-frequency correction (HFC) method reduces to a simple procedure of modifying the autocorrelation coefficients R(k), $0 \leq k \leq p$, used for LPC analysis as follows:

$$\hat{R}(k) = R(k) + \lambda V_p R(0)\psi(k), \quad 0 \leq k \leq 2,$$
$$\hat{R}(k) = R(k), \quad 3 \leq k \leq p, \tag{38}$$

where

$$\psi(0) = 3/8, \ \psi(1) = -1/4, \ \psi(2) = 1/16,$$

and V_p is the normalized error [13] of the linear predictor for the unmodified case, which is computed by solving the autocorrelation normal equations. Thus, the above HFC method requires solving the normal equations twice, once for the unmodified case and once for the modified case. (See Section 16.1.1 for a simplified HFC procedure that we have recommended for the real-time implementation.) Notice from (38) that the parameter λ controls the amount of high-frequency correction. From our experiments, we found that the choice λ=0.05 suggested in [11] was quite reasonable.

When we used the HFC procedure in the PP and PP-SQ systems using 2 bits/sample and the S-P sequence, the intensity of the beeps heard at the output was reduced, but the beeps were not eliminated. For the 4-level PP-SQ systems using the P-S sequence, adding HFC eliminated several of the discrete distortions. However, for the 3-level PP3-SQ4 system mentioned above in Section 9.3, the use of HFC was found only to reduce the number and intensity of beeps in the output speech. While the

HFC technique by itself does not solve the limit-cycle problem, it is quite effective in reducing the power gain, and yet it gives only a small drop in the S/Q ratio for frames not having excessive power gain.

9.5 Preemphasis

Preemphasizing the input speech with a filter $(1-\beta z^{-1})$ reduces its spectral dynamic range and hence reduces the power gain of A(z). The decoded speech at the receiver is deemphasized using the filter $1/(1-\gamma z^{-1})$. There are several ways of choosing β and γ: β fixed at a chosen value, β adaptively chosen as $-R(1)/R(0)$, where the R's are the autocorrelation coefficients of the input speech signal, $\gamma = \beta$, $\gamma = 0$ (no deemphasis), $\gamma < \beta$. The adaptive preemphasis method has the effect of maximally reducing the power gain, but, as it employs values of β close to 1 for most voiced sounds, the deemphasized output speech was found to have a significant amount of low-frequency roughness and rumble. We found that a fixed value of $\beta=0.4$ is a good compromise in that when used in conjunction with HFC, preemphasis eliminated all the discrete distortions in the output speech and introduced only a small amount of roughness. The choice of $\gamma<\beta$ was found to reduce the roughness slightly relative to what we perceived for $\gamma=\beta$, but the S/Q ratio was significantly lower for

$\gamma < \beta$. We used $\beta = \gamma = 0.4$ in all our subsequent experiments involving preemphasis.

Below, we summarize our experimental results on preemphasis:

1. For systems using 3-level quantization (e.g., the PP3-SQ4 system mentioned in Section 9.3), the use of HFC and preemphasis effectively eliminated the beeps and other discrete distortions. Even for 4-level systems, the use of preemphasis reduced certain discrete distortions.

2. With preemphasized input speech, the number of instabilities of a multi-tap pitch filter (used in the P-S sequence) was reduced.

3. The S/Q ratio was reduced by as much as 1 dB.

4. The output speech was perceived to be slightly rough.

5. When using preemphasis, as mentioned in Section 7.4, pole-zero noise shaping produced better speech quality than the all-pole method, which produced better speech quality than the 1-zero method.

6. Use of preemphasis in the PP-SQ-BA method (see Section 8.6) produced noticeable degradations at the output.

10. OPTIMIZATION OF CODERS FOR ERROR-FREE CHANNELS

Although the final goal of this work has been to develop a robust APC coder for use over noisy channels, initially we conducted the speech-quality optimization study for error-free channels to investigate the speech quality that the various coding methods are capable of producing at 16 kb/s, without the burden of the error protection bits. Also, we felt that parameter tradeoff relations obtained in this study could be used in narrowing the range of parameter values to investigate in the subsequent optimization study for noisy channels. The results of the study reported in this chapter and the recommendations given in Section 10.6 should be useful in the design of 16 kb/s systems for speech communication over error-free channels.

As we explained in the preceding chapters, there are several residual coding methods and noise shaping methods to choose from, and there are several parameters that affect the performance of the APC system. Important among these parameters are: input sampling rate FS, frame size (or transmission frame rate of coder parameters), number of quantization levels per residual sample, LPC order p, number of pitch-predictor taps, and parameters required by individual residual coding and noise shaping methods (e.g., number of segments). Parameters such as bandwidth w used

in noise shaping are not transmitted and, therefore, are not considered in the bit-rate tradeoff study. We reported our choice of FS=6.67 kHz in Section 2.2. When we investigated the choices p=6,8 and 10, we did not find any perceivable difference between the three cases. We decided to continue the use of p=8. The bit allocation for the various coder parameters is another dimension that affects transmission rate. As reported in earlier chapters, we chose a bit allocation that produced good results. Table 5 summarizes the bit allocation for the different parameters. Notice that individual APC systems use different subsets of parameters given in Table 5. Below, we report the results of our optimization study involving the remaining parameters, separately for each of the four coding methods. Since this study was conducted before we successfully resolved the limit-cycle problem, we used the P-S prediction sequence, without preemphasis, HFC, and limiter. Subsequent to the work reported in the last chapter, we ran the optimized P-S coder designs but with the S-P sequence. The results of this experiment are reported in Section 10.5. Informal listening tests were used to make all quality judgments reported in this chapter.

Parameter Name	Bit Allocation	Remarks
8 LARs	33	see Tables 1 and 2
Pitch	7	no quantization needed
Quantizer gain	10	for entropy coding
	6	for other methods
Pitch taps	4	for 1-tap case
	10	for 3-tap case
	16	for 5-tap case
Delta gains	2 each	see Table 4
Location parameter	2	see Section 8.5.2
Segment bit allocation	2 each	see Section 8.6

TABLE 5. Bit allocation for various coder parameters

10.1 Entropy Coding with Pitch Prediction

For entropy-coded systems, we conducted a tradeoff study involving a total of 9 coders, obtained from three values of frame size (19.5, 25.5 and 30 ms) and three conditions of pitch prediction (no pitch prediction or 0-tap, 1-tap, and 3-tap). For each of the 9 coders, we used 1-zero noise shaping and the variable-to-fixed rate conversion algorithm to adjust the quantizer step size to produce a 16 kb/s data rate. The S/Q ratios obtained for these 9 systems are given in Table 6. For each frame size, we preferred the 3-tap system over the 0-tap and

1-tap systems. For the three 3-tap systems, the average number of bits/sample used was found to be 1.9, 2.01 and 2.07, respectively. Two 3-tap systems, one with 25.5 ms frame size and the other with 30 ms frame size, were found to produce the best overall speech quality. We investigated these two systems further, using pole-zero noise shaping with various values of the bandwidth parameter w. We found that the choice w=800 Hz produced the best perceptual result, which was much better than what we obtained with 1-zero noise shaping. The two 3-tap systems with 800 Hz pole-zero noise shaping produced S/Q ratios of about 19.0 dB (25.5 ms) and 19.3 dB (30 ms). We found the second 3-tap system (30 ms) to produce marginally better speech quality than the first (25.5 ms). For comparisons with other optimized systems, therefore, we chose the EC-PP3 system with a frame size of 30 ms.

	Frame Size(ms)		
No. of Pitch Taps	19.5	25.5	30
0	19.5	19.9	20.0
1	19.7	20.8	21.2
3	20.0	21.2	21.6

TABLE 6. S/Q ratios for the 9 entropy-coding APC systems, all using 1-zero noise shaping

10.2 Pitch Prediction and Segmented Quantization

In Section 8.3.3, we reported the results of an experiment involving 9 PP-SQ systems (three values each of the number of pitch taps and the number of segments), all using a frame size of 25.5 ms. Although these systems did not have the same bit rate, the individual bit rates were close to 16 kb/s. The two systems PP3-SQ5 and PP3-SQ10 produced the best overall speech quality. We then conducted a tradeoff study involving the three 16 kb/s systems: PP3-SQ5 (25.5-ms frame), PP3-SQ10 (29.25-ms frame), and PP1-SQ4 (22.5-ms frame). Comparative speech quality evaluations indicated that the PP3-SQ10 system was the best. This system produced an average S/Q ratio of 18.2 dB with the use of the 1-zero noise shaping. The other methods of noise shaping did not produce any perceivable speech quality improvement.

10.3 Pitch-Adaptive Coding With Pitch Prediction and Segmented Quantization

For pitch-adaptive coding, we considered three 16 kb/s systems trading off frame size, average number of bits/sample, and number of pitch taps: (1) PA4-PP3-SQ3, bit allocation = {3,2,1,2} with the average being 2 bits, and 25.5-ms frame; (2) PA10-PP1-SQ3, bit allocation {3,3,3,2,2,2,2,1,1,2} with the

average being 2.1 bits, and 30-ms frame; and (3) PA5-PP3-SQ3, bit allocation = {3,2,1,1,2} with the average being 1.8 bits, and 19.5-ms frame. Since we combined segments with the same bit allocation for the purpose of segmented quantization, we transmitted only 3 segment gains in each of these three coders (see Section 8.5.3). Informal listening tests indicated that the first coder produced the best overall speech quality. This coder produced an average S/Q ratio of 18.8 dB with the use of 1-zero noise shaping.

10.4 Segmented Quantization With Bit Allocation and Pitch Prediction

For the PP-SQ-BA system, we investigated frame sizes less than 27 ms (see Section 8.6.2), number of pitch taps equal to 1, 3, or 5, number of segments up to 10, and various segment bit allocations. From this investigation, we found that the PP5-SQ10-BA system with a frame size of 25.5 ms and bits/sample from the set {0,1,2,3} produced the best overall speech quality. This system used an average number of bits/sample B_0=1.9 and produced an average S/Q ratio of about 14.8 dB with 1-zero noise shaping.

10.5 The S-P Prediction Sequence

We re-investigated the above-described optimized coders using the S-P prediction sequence and the APC-HF configuration. For the EC-PP3 and the PP5-SQ10-BA systems, we used HFC, and a limiter. For the PP3-SQ10 and the PA4-PP3-SQ3 systems, we used preemphasis, HFC, and a limiter. In all the cases, we used pole-zero noise shaping.

By and large we obtained about the same speech-quality performance from the S-P sequence as was observed using the P-S sequence. However, there are three noteworthy differences between the systems resulting from the two prediction sequences. These differences, reported in the preceding chapters, are summarized below:

1. Since the multi-tap pitch filter was found to be stable for the S-P sequence, the switched prediction method, which requires checking the stability of the pitch filter and which is necessary for the P-S sequence (see Section 8.2), is not needed for the S-P sequence.

2. With pole-zero noise shaping, the S-P sequence leads to a simple implementation, since $A(z)B(z)$ reduces to $A(z/\alpha)$ (see Section 7.3).

3. With the use of the S-P sequence and the APC-HF configuration, a limiter can be effectively used in the noise-feedback path as a worthwhile precaution (see Section 9.3).

10.6 Comparative Evaluation and Recommendations

We conducted informal listening tests to compare the four optimized coders: (1) EC-PP3, (2) PP3-SQ10, (3) PA4-PP3-SQ3, and (4) PP5-SQ10-BA. The output speech from the systems (2)-(4) was slightly better than from the system (1), but occasionally it contained low-level discrete distortions. In contrast, the entropy-coded system produced relatively smooth speech. While the perceived speech-quality differences among the four systems were small, three subjects, upon careful listening, rated them in the following order from best to worst: EC-PP3, PA4-PP3-SQ3, PP5-SQ10-BA, PP3-SQ10. From the viewpoint of minimizing computational complexity, the ordering of these systems is just the reverse of the above order, with EC-PP3 being the most complex system (because of its variable-to-fixed rate conversion).

Comparing the output of each of the four systems against the input speech using high-quality headphones with good low- and

high-frequency response, we found that the EC-PP3 system produced speech closest, but not identical (or transparent), to the natural speech.

For 16 kb/s speech communication over error-free channels, if computational complexity is not an issue, we recommend the use of the EC-PP3 system. As for the specific configuration to use, from the observations given in Section 10.5, we recommend the implementation of this system using the S-P prediction sequence, APC-HF configuration, high-frequency correction, and a limiter. For noisy-channel applications, as will be seen in Chapter 12, both the recommendations have to be modified.

11. OPTIMIZATION OF ERROR-PROTECTED CODERS

In this chapter, we present the results of our parameter optimization study, performed in the absence of channel errors, of 16 kb/s APC systems in which bits have been allocated for the error protection of coder parameters. The performance of the error-protected APC systems in 1% channel error is the topic of the next chapter. The objective of our work reported in this and the next chapter was to develop a robust 16 kb/s APC coder to operate over channels with bit-error rates of up to 1%. To meet this objective, we experimentally optimized (1) the tradeoff between the voice data rate and the error-protection data rate, and (2) the amount of error protection for individual transmission parameters. In this chapter, we present several error-protected APC systems for investigating the tradeoff (1) above. We did not protect the coded APC residual. To partially protect important parameters of the coder, we used the Hamming (7,4) code, which protects 4 data bits by adding 3 parity bits; this code detects and corrects all single bit-errors in the resulting 7-bit codeword. In the APC systems reported below, the number of protected bits per frame varies from 28 to 68. Our choice of this moderate to substantial protection of the side-information data was based on our previous experience [3]. We conducted informal listening tests to compare these error-

protected APC systems in the absence of channel errors. The results are presented below separately for the three residual coding methods discussed in Chapter 8. We did not include the pitch-adaptive coding method in the channel-error study, since we felt that its implementation on the MAP would be extremely difficult. For the systems described in this chapter, we used the P-S prediction sequence.

11.1 Entropy Coding with Pitch Prediction

To study the tradeoff between frame size and the number of pitch-filter taps, we considered four systems: EC, EC-PP1, EC-PP3, and EC-PP5. The details of these systems are given in Table 7. Notice that Items 3 and 4 given in the table are both EC-PP3 systems using different noise shaping methods. We used high-frequency correction with λ=0.05 for all five systems in Table 7. Notice that each of these systems protects a relatively large number of parameter bits. To obtain a 16 kb/s data rate, we used the variable-to-fixed rate conversion algorithm described in Section 8.4.2. Table 7 also gives the average bits/sample used by the individual coders. Systems 1-3 in Table 7 use 1-zero noise shaping. Among these three systems, we found that the EC-PP3 system (System 3) produced the best overall speech quality. When we used 800 Hz pole-zero noise shaping for the EC-PP3

system, the S/Q ratio dropped from 20.1 dB (for System 3) to 17.7 dB (for System 4), but the perceived background noise was significantly reduced relative to the 1-zero case. Comparing the 3-tap system (System 4) with the 5-tap system (System 5), we found that the former system produced slightly better speech quality.

No.	System	Frame Size(ms)	Parameter Bits Total	Parameter Bits Protected	Avg.Bits Per Sample	Noise Shaping	S/Q Ratio (dB)
1	EC	25.5	43	32	1.97	1-zero	18.6
2	EC-PP1	27.0	54	40	1.89	1-zero	19.5
3	EC-PP3	30.0	60	44	1.90	1-zero	20.1
4	EC-PP3	30.0	60	44	1.90	pole-zero (800 Hz)	17.7
5	EC-PP5	30.0	66	48	1.86	pole-zero (800 Hz)	17.6

TABLE 7. 16 kb/s error-protected entropy-coding systems

11.2 Pitch Prediction and Segmented Quantization

For the tradeoff study involving LPC order, number of pitch-filter taps, frame size, number of quantizer levels per residual sample, number of segments, and number of bits protected, we considered eight 16 kb/s PP-SQ systems given in Table 8. For all eight systems, we used preemphasis (β=0.4), high-frequency

correction (λ=0.05), and pole-zero noise shaping (w=800 Hz). The first two systems use 3-level quantization and provide substantial protection of parameters as in the entropy-coding systems considered above. Of these two systems, the PP5-SQ10 system was found to produce speech with more clarity. Notice that these two 3-level systems require block-coding of residual samples; five residual samples are coded using 8 bits. Therefore, a single bit-error causes wrong decoding of five samples. From our experience with the design of the 9.6 kb/s baseband coder [3], we anticipated that such a block-coding method would result in poor channel-error performance. Therefore, we considered six systems (Systems 3-8 in Table 8) using 4-level quantization. In choosing these six systems, we varied the different coder parameters and provided, for the ratio of the number of protected bits to the total number of parameter bits per frame, a range of values from 28/58 (System 3) to 44/56 (System 8). Of these six systems, we found that the PP1-SQ4 system had the highest level of background noise. Comparing the four systems with 33.75 ms frame size, we noted that the speech from the PP3-SQ2 system (System 4) was somewhat rougher than from the other three systems, and that these latter three systems (Systems 5,6, and 8) produced about the same speech quality. The speech from System 7 lacked the clarity produced by the other systems with smaller frame sizes, which indicated that the update

rate for the parameters in that system was not adequate. Therefore, Systems 5, 6, and 8 are our preferred choices of the 4-level systems. Of these three, System 8, which is the 6-pole PP3-SQ3 system, provides the largest protection of parameter bits. When we compared this 4-level system with the 3-level system PP5-SQ10 (System 2), we found that the 4-level system yielded slightly more natural-sounding speech than the 3-level system. We note that the low S/Q ratio values for the systems in Table 8 are primarily because of the noise shaping used. (Without any noise shaping, the S/Q ratio of System 8 was found to be 17.3 dB.)

No.	System	LPC Order (p)	Frame Size (ms)	No. of Levels/ Sample	Parameter Bits Total	Parameter Bits Protected	S/Q Ratio (dB)
1	PP3-SQ4	8	19.5	3	64	52	13.9
2	PP5-SQ10	8	25.5	3	82	68	12.9
3	PP1-SQ4	8	30.0	4	58	28	13.2
4	PP3-SQ2	8	33.75	4	60	36	13.4
5	PP3-SQ3	8	33.75	4	62	36	13.6
6	PP3-SQ4	8	33.75	4	64	32	13.8
7	PP3-SQ3	8	36.6	4	62	44	12.9
8	PP3-SQ3	6	33.75	4	56	44	13.8

TABLE 8. 16 kb/s error-protected PP-SQ systems

11.3 Segmented Quantization With Bit Allocation and Pitch Prediction

The results reported above for the PP-SQ system simplified the problem of choosing the PP-SQ-BA system(s) appropriate for the channel-error study. Recall from Section 8.6.2 that frame sizes larger than 27 ms lead to perceivable distortions in the output speech of the PP-SQ-BA system. Based on these considerations, we chose for the channel-error study the PP3-SQ10-BA system with frame size = 25.5 ms, LPC order p = 6, segment bit allocation {0,1,2,3}, and B_0=average bits/sample=1.7. This system protects 64 bits out of a total of 70 bits/frame of side-information data. Using high-frequency correction (λ=0.05), no preemphasis, and 400 Hz pole-zero noise shaping, we obtained an average S/Q ratio of 13.8 dB.

11.4 Comparative Evaluation

We compared the three error-protected systems in the absence of channel bit-errors: EC-PP3, PP3-SQ3 (System 8 in Table 8), and PP3-SQ10-BA. We found different types of distortions in the output speech from the three systems. The EC-PP system had the highest level of background noise, but it produced a more-pleasing smooth speech. The PP-SQ and PP-SQ-BA systems had

"choppy" background noise, but their output speech was perceived as more natural than that of the EC-PP system. The PP-SQ-BA system had a "scratchy" quality. Despite these differences, we felt that the three systems were of roughly equivalent overall quality. The final choice of a robust APC system should be determined only after comparing the channel-error performance of these three systems and others described above in this chapter.

12. EVALUATION OF ERROR-PROTECTED CODERS IN 1% CHANNEL ERROR

One of the requirements of this project has been to design a robust 16 kb/s APC system that tolerates adequately channel bit-error rates of up to 1%. We used the following engineering criterion suggested by the COTR as a specific measure of the extent of robustness required of the final APC system design: The speech quality of the error-protected 16 kb/s coder at 1% channel error should be about the same or better than the speech quality of the same coder when it is operated without error protection in 0.1% channel error. Notice that the unprotected, engineering-criterion system will have a bit rate less than 16 kb/s, since it is obtained by discarding the error-protection bits of a 16 kb/s coder.

In the last chapter, we considered the tradeoff between the voice data rate and the error-protection data rate. In this chapter, we present the results of our work on the following issues: (1) distribution of the allocated error-protection bits among individual transmission parameters (Sections 12.2-12.4); (2) selection of the coder having the best channel-error performance for each of the three coding methods considered (Sections 12.2-12.4); (3) effect of the prediction sequence on the channel-error performance of a coder (Section 12.5); (4)

comparative evaluation of the optimized coders and choice of the most robust 16 kb/s coder (Section 12.6); (5) effect of using the so-called folded binary code for encoding the residual (Section 12.7); and (6) investigation of the performance of the robust system over higher-error-rate channels (Section 12.8). Before we proceed to present the results on these topics, we provide in Section 12.1 a brief description of our channel-error simulation.

12.1 Channel-Error Simulation

In our simulation, we used the binary symmetric channel in which independent, identically distributed random errors are introduced into the transmitted bit stream. A bit error simply changes the state of the bit from 0 to 1 or from 1 to 0. Our simulation system permits the user to vary both the bit-error rate and the amount of error protection separately for each parameter. We used this feature to (1) examine how each transmitted parameter, when subjected to 1% channel error, would independently affect the output speech and (2) investigate, as a diagnostic tool, the cause of the perceived distortions in the output speech. In general, we found that channel bit-errors on the (unprotected) APC residual samples caused the output speech to have a continuously rough or "raspy" character and a reverberant quality. In contrast, uncorrected bit-errors on the

side-information data caused discrete distortions such as pops and clicks in the speech. Specific results are given in the following sections.

12.2 Entropy Coding with Pitch Prediction

For the entropy-coding systems, as mentioned in Section 8.4.1, we used the self-synchronizing code with codewords 0,10,110,etc. Clearly, one bit-error in a codeword can cause one of two decoding errors: splitting a sample into two samples, or merging two samples into one. This will cause the total number of decoded samples in a frame to be larger or smaller than the desired, fixed number. In view of a requirement stemming from the real-time implementation on the MAP, we chose to discard samples at the end of the frame if a larger number of samples were decoded and to append zero samples to the end of the frame if a smaller number of samples were decoded.

Initially, we conducted our experiments using the EC-PP3 system reported in Section 11.1, to determine the amount of protection required by the individual parameters and to understand the source of each of the different distortions perceived in the output speech. We found that the amount and the specific distribution of error protection of parameter data of

the EC-PP3 system given in Table 9 was quite effective in coping with 1% channel bit-errors; when the residual samples were not subjected to channel error, the performance of this coder was found to be approximately invariant as the bit-error rate on the parameters was varied from 0% to 1%. However, with the introduction of 1% bit-errors on the residual samples, we found that the output speech had a "ringing" or reverberant quality.

In our subsequent experiments, we compared the 1% channel-error performance of the entropy-coding systems reported in Section 11.1. The specific error-protection allocations used for these systems are given in Table 9. Listening tests showed that the ringing or reverberant quality was substantially worse for the EC-PP1 system than for the EC-PP3 system. The EC-PP5 system produced slightly less reverberant quality than the EC-PP3 system did but caused perceivable distortions such as pops. The EC system (which does not use pitch prediction) did not produce a reverberant quality but exhibited a continuously rough or raspy character, which degraded the speech quality substantially. The output speech of the EC system sounded almost like whispered speech, without proper pitch periodicity. From the results of these tests, we make the following two conclusions: (1) Although using pitch prediction causes the output speech to sound reverberant, it yields significantly better overall speech

Parameter		EC	EC-PP1	EC-PP3	EC-PP5
Quantizer gain		10(9)	10(9)	10(9)	10(9)
Pitch			7(6)	7(6)	7(6)
Taps	c(M-2)	--	--	--	3(2)
	c(M-1)	--	--	3(2)	3(2)
	c(M)	--	4(3)	4(3)	4(3)
	c(M+1)	--	--	3(2)	3(2)
	c(M+2)	--	--	--	3(2)
Log Area Ratios	1	6(5)	6(5)	6(5)	6(5)
	2	5(4)	5(4)	5(4)	5(4)
	3	4(3)	4(3)	4(3)	4(3)
	4	4(3)	4(3)	4(3)	4(3)
	5	4(3)	4(3)	4(3)	4(3)
	6	4(3)	4(2)	4(2)	4(2)
	7	3(2)	3(2)	3(2)	3(2)
	8	3	3	3	3
Error Protection: Total protected		(32)	(40)	(44)	(48)
Cost		24	30	33	36
Sync		1	1	1	1
Total bits/frame		68	85	94	103

TABLE 9. Error-protection allocations used for four entropy-coding systems. Numbers given within parentheses indicate the number of the most significant bits protected.

quality than the scheme without pitch prediction; and (2) 3-tap pitch prediction produces the best overall speech quality. Combining these results with the results reported in Section 11.1 for the 0% error case, we chose the EC-PP3 system as the most robust 16 kb/s entropy-coding APC system.

12.3 Pitch Prediction and Segmented Quantization

Although we tested in channel error several of the PP-SQ coders described in Section 11.2, we present below the results for the two interesting systems: 3-level PP5-SQ10 (System 2 in Table 8) and 4-level PP3-SQ3 (System 8 in Table 8). Both systems provide for substantial error protection of side-information data. Table 10 gives the amount of error protection we used for individual parameters in each of the two systems. We obtained this error protection allocation among parameters through several experiments. We found that full protection of the 2-bit segment gains was necessary to reduce unpleasant pops and clicks.

Output speech from the PP5-SQ10 system operating in 1% channel error contained discrete distortions and had a continuously rough and reverberant quality. To examine the extent to which these quality degradations were caused by the block-coding of the residual samples used in this 3-level system,

Parameter			PP5-SQ10	PP3-SQ3	PP3-SQ10-BA
Quantizer gain			6(6)	6(6)	6(6)
Pitch			7(7)	7(7)	7(7)
Taps		c(M-2)	3(2)	--	--
		c(M-1)	3(2)	3(2)	3(2)
		c(M)	4(3)	4(3)	4(3)
		c(M+1)	3(2)	3(2)	3(2)
		c(M+2)	3(2)	--	--
Segment	Delta gains		10 x 2(20)	3 x 2(6)	--
	Bit allocation		--	--	10 x 2(20)
Log Area Ratios		1	6(5)	6(5)	6(6)
		2	5(4)	5(4)	5(5)
		3	4(3)	4(3)	4(4)
		4	4(3)	4(2)	4(3)
		5	4(3)	4(2)	4(3)
		6	4(3)	4(2)	4(3)
		7	3(2)	--	--
		8	3(1)	--	--
Error protection: Total protected			(68)	(44)	(64)
Cost			51	33	48
Sync			1	1	1
Total bits/frame			134	90	119

TABLE 10. Error-protection allocation used for two PP-SQ systems and one PP-SQ-BA system. Numbers given within parentheses indicate the number of the most significant bits protected.

we simulated the same system without block coding (i.e., we used 2 bits to encode the output of the 3-level quantizer). It was noted that the output speech from this latter system was less rough and reverberant. Also, several discrete distortions, observed in the block-coded case, were removed.

The 4-level PP3-SQ3 system, on the other hand, produced speech that was less reverberant and substantially less rough than the PP5-SQ10 system. Any significant reduction in the side-information protection for the PP3-SQ3 system was found to increase the number and the intensity of the perceived discrete distortions in the output speech. Also, recalling from Section 11.2, this PP3-SQ3 system performed at least as well as any other PP-SQ system that we tested in 0% channel error. Therefore, we chose the 4-level, 6-pole PP3-SQ3 system as the most robust PP-SQ system.

12.4 Segmented Quantization with Bit Allocation and Pitch Prediction

For the PP3-SQ10-BA system described in Section 11.3, we protected fully all ten 2-bit codes representing the segment bit allocations, since errors in these codes cause wrong decoding of some or all of the residual samples of a frame. The error protection we chose for other parameters is given in Table 10.

The output speech from this coder in 1% channel error was found to have several discrete distortions and be somewhat reverberant.

12.5 Effect of Prediction Sequence on Channel-Error Performance

The APC systems considered in the above-described channel-error study used the P-S prediction sequence. We investigated the channel-error performance of some of these systems using the S-P prediction sequence. The results of this investigation are given in Section 12.5.1. In an attempt to improve the inferior channel-error performance caused by the S-P sequence, we incorporated at the receiver means of smoothing the decoded residual samples. The results of this study are given in Section 12.5.2.

12.5.1 The S-P Prediction Sequence

For the S-P prediction sequence, we used the APC-HF configuration with a limiter in the noise-feedback path. Preemphasis and high-frequency correction were used in the same way as with the P-S prediction sequence (see Chapter 11). Since we found that using the autocorrelation method of pitch-tap computation yielded fewer discrete distortions in 1% channel error than using the covariance method, we used the

autocorrelation method in the subsequent experiments. Listening tests comparing the 1% channel-error performance produced by the two prediction sequences for otherwise identical APC systems showed that the speech from the S-P system was slightly less reverberant than from the P-S system, but it contained objectionable discrete distortions. The overall speech quality from the S-P sequence was found to be inferior to that from the P-S sequence.

12.5.2 Receiver Smoothing of the Decoded Residual

In an attempt to improve the channel-error performance produced by the S-P sequence, we investigated two modifications to the APC system. Both modifications were motivated by the following considerations. The channel bit-errors may be thought of as an additive random noise corrupting the transmitted bit stream. For the P-S prediction sequence, which leads to a good channel-error performance as we noted above, this additive noise is "smoothed" by the spectral filter $1/A(z)$ before it is processed by the pitch filter $1/C(z)$. Although the pitch filter propagates a bit-error in a periodic manner and with a relatively long time constant, the effect of this smoothing provided by the spectral filter may be responsible for the observed good channel-error performance of the coder using the P-S sequence. With the

S-P sequence, the additive noise mentioned above is processed directly by the pitch filter. This may be responsible for the resulting poor channel-error performance.

First, we reversed the order of the pitch and spectral filters in the receiver of the S-P system so that the receiver processed the channel errors on the residual samples the same way as in the P-S system. Although this change should introduce some additional distortion in the output speech for error-free channels, we hoped that the possible improvement in the coder's channel-error performance might outweigh that bad effect. However, our tests showed that the output speech contained severe distortions and frequently had reverberant quality both in the presence and in the absence of channel bit-errors.

Second, we investigated the effect of inserting a smoothing operation in the receiver, to smooth the decoded residual samples [32]. We investigated two types of smoothing: linear (average) and non-linear (median). We used a 3-point average smoother and both 3-point and 5-point median smoothers. In the presence of channel bit-errors, smoothing reduced some of the discrete distortions in the speech. In this regard, average smoothing was preferred over median smoothing. However, both types of smoothing introduced substantial smearing and muffling of the speech, thus lowering the overall speech quality significantly.

Since neither of the above two modifications improved the channel-error performance of the coder using the S-P prediction sequence, we chose to use the P-S prediction sequence (without smoothing) in our optimized design of the robust APC coder.

12.6 Comparative Evaluation and Recommendations

We compared the three optimized systems, EC-PP3, 4-level PP3-SQ3, and PP3-SQ10-BA, in 1% channel error. A general comment should be made regarding comparisons of different systems in the presence of channel bit-errors. The perceived quality of speech transmitted over an errorful channel depends on the particular realization of the random process causing the channel bit-errors. In comparing two systems, therefore, the speech-quality judgments should be made over a large amount of speech, instead of on a sentence-by-sentence basis. Following this method and using the 12 sentences from the high-quality data base, we found the PP-SQ and PP-SQ-BA systems to be very similar in overall quality. The PP-SQ-BA system produced more discrete distortions and less reverberant quality than the PP-SQ system did. The EC-PP system produced much more reverberant speech and was clearly inferior to the other two systems.

We then tested each of the three systems to check if it

satisfied the engineering channel-error criterion mentioned at the beginning of this chapter. For the EC-PP system, the unprotected system operating in 0.1% channel error produced less reverberant and better overall speech quality than the error-protected system did in 1% channel error. For both the PP-SQ and PP-SQ-BA systems, the protected and unprotected systems yielded roughly the same overall speech quality.

In view of the performance equivalence of the PP-SQ and PP-SQ-BA systems in both error-free and errorful channels, we have chosen the PP-SQ system for real-time implementation. We feel that this is the safer choice, since the PP-SQ system was found to be more robust in the sense that it performed in a more consistent and uniform manner over individual sentences than the PP-SQ-BA system did. The reason for this difference is that while channel bit-errors on the transmitted bit-allocation parameters of the BA scheme can cause erroneous decoding of a part of the residual samples, proper decoding is always ensured in the PP-SQ system. It is therefore reasonable to expect that the PP-SQ system will continue to perform well for speech utterances other than those we used in our study. The PP-SQ system is also less complex and uses a larger frame size, simplifying the requirements on the real-time computation speed.

Finally, we compared the output speech from the chosen PP-SQ

system obtained for two cases: 0% and 1% channel error. Quite impressively, the degradations caused by channel error were found to be perceivable but small; the degradations were in the form of roughness and a slightly reverberant quality.

12.7 Folded Binary Code

In all our channel-error simulations reported thus far, we used the natural binary code (NBC) to encode the quantized residual samples. To improve the robustness of the PP-SQ system further, we investigated the use of an alternate encoding method called the folded binary code (FBC). For FBC, the most significant bit gives polarity information; the remaining bits represent the sample magnitude in natural binary code. Figure 14 illustrates the difference between the two encoding methods, for the 4-level quantizer. For NBC, going from the most negative to the most positive level, the 4 codes are: 00,01,10, and 11. For FBC, the 4 codes are: 01,00,10, and 11. We note that the word "folded" comes from the fact that the codes for the first two levels of NBC have been reversed in FBC. To show when and how FBC yields better performance than NBC, let us assume that only single bit-errors occur within the 2-bit residual code. Referring to Fig. 14, a bit-error causing 00 to be received as 10 results in a decoding error of 2.238 (=0.419+1.829; see Fig. 14)

for NBC, but results in a decoding error of only 0.838 (=0.419+0.419) for FBC, both decoding errors computed for the Laplacian quantizer shown in Fig. 14. It can be shown that for single bit-errors in the residual code, the mean-square decoding error is the same for all the four levels and equal to 3.52 for NBC. Using FBC reduces the mean-square decoding error of each of the two inner levels to 1.35 at the expense of increasing the decoding error of each of the two outer levels to 7.68. Therefore, if the combined probability of occurrence of the two inner levels is greater than 0.5, which is the case in the PP3-SQ3 system, then using FBC produces a smaller overall mean-square decoding error than using NBC.

When we used FBC in the PP-SQ system for encoding the output levels of a 4-level Laplacian quantizer, we obtained perceivable improvements in the output speech quality in the form of a reduction in both the reverberant quality and the raspy character.

12.8 Performance Over Higher Error-Rate Channels

Since certain Department of Defense applications may have the need to operate 16 kb/s APC coders over channels having error rates higher than 1%, the COTR suggested the testing of the

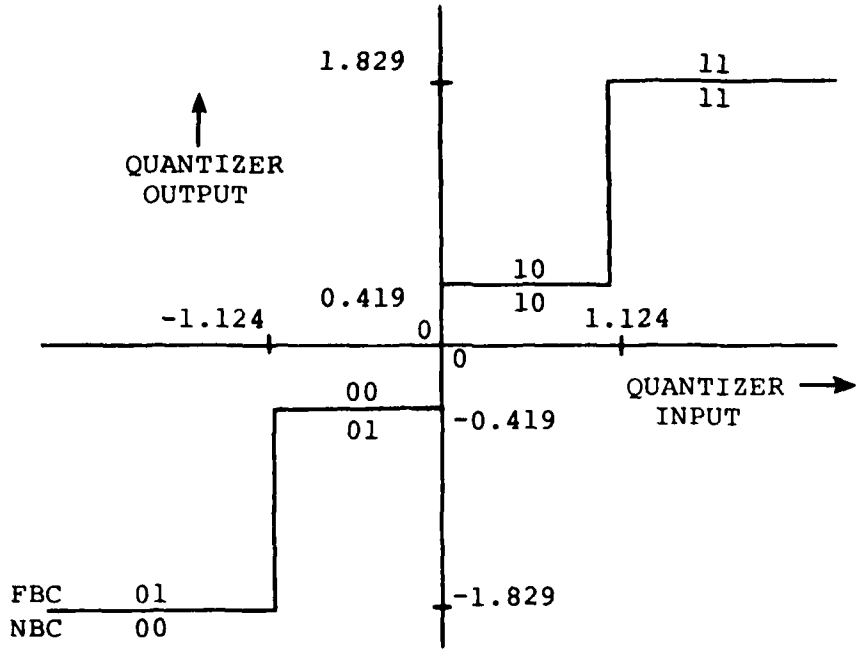


FIG. 14. Illustration of the difference between natural binary code (NBC) and folded binary code (FBC), for a 4-level Laplacian nonuniform quantizer.

chosen robust coder in channel error rates of 2% to 5%. Of course, the design requirement of this project was to achieve a robust performance only for channel error rates of up to 1%. For the higher error rate channels, we obtained the following two results: (1) the output speech intelligibility is satisfactory for 2% bit-errors; and (2) the output speech is not acceptable for 3% and higher bit-error rates, with loud pops and frequent drop-outs of entire words. The reason for this poor performance may be that the effectiveness of the Hamming (7,4) code used in

the APC coder breaks down at these high error rates. Mathematically, it can be shown that the average error rate of the decoded bits is about 0.2% for 1% channel error and about 3% for 4% channel error. We conducted another experiment to determine if the poor performance of the coder was caused by the breakdown of the effectiveness of the Hamming code or by the effect of the channel errors on the unprotected residual signal. Using the FORTRAN simulation of the PP-SQ coder (see Appendix B), we simulated a coder in which only the residual signal was exposed to channel errors, and we processed six sentences at error rates of 3%, 4%, and 5%. The processed speech was quite intelligible even at the 5% error rate. With respect to speech quality, the processed speech sounded generally more raspy as the error rate was increased from 3% to 5%, and the speech from female talkers exhibited more reverberant quality. The perceptual effect of channel errors on only the residual signal seemed similar to that of the quantization noise. From the results of this experiment, we conclude that for an application involving high error-rate channels, more powerful codes than the Hamming (7,4) code and a larger amount of error protection of coder parameters than we have used in the above PP-SQ system are required to yield satisfactory speech intelligibility.

13. ACOUSTIC BACKGROUND NOISE

We tested the performance of the optimized, robust PP3-SQ3 coder for input speech corrupted by one of two acoustic background noise types: office noise (about 60 dB SPL) or ABCP noise (about 90 dB SPL). For these tests, we used sentences from the office-noise data base and from the ABCP data base described in Section 2.3. For the office-noise case, the coder produced output speech with very good quality. For the ABCP noise, the output speech of the coder was found to have good quality and intelligibility. We noted that the output speech quality in the first case was found to be closer to the input speech quality than we observed for the case using the high-quality speech input, and the input-output quality comparison was even closer for the ABCP data base.

14. TANDEMING WITH LPC-10

14.1 Simulation of LPC-10

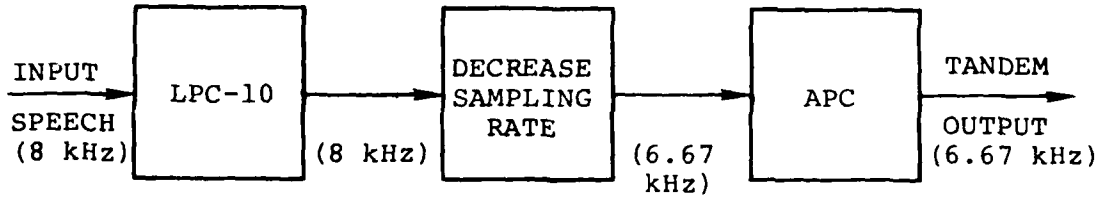
We installed on our DECSystem-20 the PDP-11 FORTRAN implementation of 2.4 kb/s the LPC-10 (version 42) coder [27]. The process of bringing up the LPC-10 coder simulation on our computer involved, among other things, the following two tasks: modification of the input/output sections to accept our formatted speech waveform files and modification to the subroutine "CHANL", which assumes a 16-bit wordlength, to operate properly on our 36-bit computer. The output speech from this implementation of LPC-10 was compared against the audio tape recording of the output from the original PDP-11 implementation of LPC-10. This comparative evaluation and subsequent consultations with the DoD agency that supplied the LPC-10 program clearly indicated that our implementation of LPC-10 was functioning correctly.

Before we present the detailed results of our investigation of the tandem link between APC and LPC-10, we point out that this tandem produced satisfactory performance in either direction unlike the tandem connection between 16 kb/s CVSD coder and LPC-10 [34].

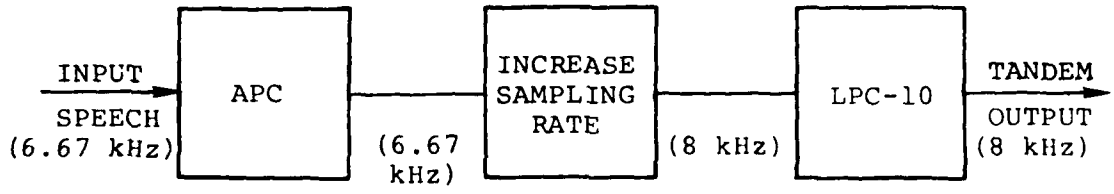
14.2 LPC-APC Tandem

Since the LPC-10 coder employs a 8 kHz sampling rate and our FORTRAN simulation of the APC coder employs a 6.67 kHz sampling rate, the digital tandem interface between the two coders must have provisions for changing the sampling rate. For the LPC-APC tandem, shown in Fig. 15(a), the interface must reduce the sampling rate from 8 kHz to 6.67 kHz; we performed this sampling rate reduction by 5:1 interpolation followed by 6:1 decimation. We used high-order FIR lowpass filters for the operations of interpolation and decimation.

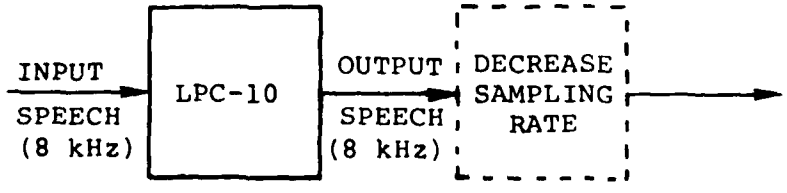
Since the LPC-10 coder uses, as excitation for voiced sounds, a stored version of the impulse response of an allpass filter, its output is not expected to have a high peak factor (peak-to-rms ratio). Signals with a high peak factor may in general increase the clipping errors in an APC system and hence introduce additional distortions in the APC speech. However, the optimized coder design obtained in this work employs 3-tap pitch prediction and 3-segment segmented quantization to track the varying signal amplitudes. In fact, the output speech from the LPC-APC tandem had about the same perceived speech quality as the LPC-10 speech band-limited to 3.33 kHz (shown in dashed lines in Fig. 15(c)). The single-link LPC-10 output was slightly more



(a) LPC-APC tandem



(b) APC-LPC tandem



(c) Single LPC-10 link

FIG. 15. Tandem operation of APC and LPC-10 coders.

"crisp" than the tandem output. Since the LPC-10 speech has energy in the frequency range 3.33-4.0 kHz, potentially it may have a slightly higher speech intelligibility than the speech from the tandem link. We did not conduct any formal speech intelligibility tests.

14.3 APC-LPC Tandem

Because of the difference in the sampling rates of the two coders, the tandem interface must increase the sampling rate from 6.67 kHz to 8 kHz, as shown in Fig. 15(b). We achieved this sampling rate increase by 6:1 interpolation followed by 5:1 decimation.

The output speech from the PP3-SQ3 APC coder has granular noise and some clipping noise. One effect of noise in speech is to reduce its short-term spectral dynamic range. The linear prediction analysis of LPC-10 on APC speech would, therefore, overestimate formant bandwidths, and the resulting speech would in general sound buzzy and be of lower quality than speech from a single LPC-10 link.

In our testing of the APC-LPC tandem, we obtained results very similar to the ones reported above for the LPC-APC tandem, with one difference. The APC-LPC tandem produced output that was

slightly inferior to the output from the LPC-APC tandem. Let us consider ways of improving the performance of the APC-LPC tandem. One method that we suggested in our proposal [1] is to enhance the APC speech through a spectral subtraction method given in [33], before processing through the LPC-10 coder. This method should reduce the distortions caused by the noise in the APC speech. However, we feel that a more serious source of quality problem is at the digital tandem interface. The speech coming out of the interface has a spectrum with a sharp amplitude change (discontinuity) at about 3.33 kHz and with very small amplitudes in a region just below 4 kHz. The subsequent LPC analysis would unduly "spend" some of its resources in attempting to model the spectral discontinuity. Said another way, the LPC analysis makes effective use of fewer than 10 coefficients (which is the number of poles used in LPC-10, for voiced frames). A reasonable solution to this problem would be to "fill in" the spectral gap between 3.33 kHz and 4 kHz using, for example, the high-frequency correction method [11] that we have described in Section 9.4. This spectral correction can be done as part of the interface or as a user-selectable option within the LPC-10 coder. In the latter case, the HFC method can be implemented by simply modifying the elements of the covariance matrix as suggested in [11].

15. 16 KB/S BASEBAND CODERS

As we mentioned in Chapter 2, the baseband coder is one of the three types of coders capable of producing toll quality speech at 16 kb/s. Since we recently designed and implemented on the MAP-300 a 9.6 kb/s BBC system as part of a DCA contract [3], we proposed to extend this design to the 16 kb/s case and compare the performance of the resulting BBC system with that of the optimized, PP3-SQ3 APC system. The results of this work are reported below.

Based on our previous experience [3], we chose to investigate two 16 kb/s baseband coder designs. Both coders use a baseband width of 1.67 kHz, encode the baseband residual using an APC coder with pitch prediction and no spectral prediction, and perform high-frequency regeneration at the receiver using the perturbed spectral folding method [3]. The two 2-band BBC systems are defined in terms of their parameter values, as follows:

System 1

> 21 ms frame size, 11 quantization levels per baseband residual sample, 3-tap pitch prediction for the baseband APC coder, and 44 of 55 parameter bits protected against channel error.

System 2

27 ms frame size, 16 quantization levels per baseband residual sample, 1-tap pitch prediction for the baseband APC coder, and 28 of 49 parameter bits protected against channel error.

In the error-free channel, System 2 had less background noise and was judged to be of higher quality than System 1. We compared System 2 with the optimized fullband APC coder (PP3-SQ3). The fullband APC coder had noticeably more background noise, particularly at low frequencies, but had better overall speech quality because of the unnatural high-frequency distortions produced by the baseband coder. We repeated the same comparison in the presence of 1% channel error. In this test, the fullband APC coder speech quality was clearly superior to that of the baseband coder.

16. OPTIMIZED, ROBUST 16 KB/S APC CODER

In this chapter, we report the results of further work on the PP3-SQ3 coder that we chose as the most robust coder. This work consisted of 1) making some performance-preserving simplifications to the coder design for facilitating real-time implementation on the MAP-300 and 2) providing some refinements to the coder design, to improve the coder performance further. Then, we summarize the details of the final design of the coder and introduce Appendices A-C, which contain a detailed specification and FORTRAN simulation of the coder. Finally, we present and discuss the results of our tests on the real-time implementation of the coder.

16.1 Simplifications for Real-Time Implementation

16.1.1 High-Frequency Correction

Recall from Section 9.4 and Eq. (38) that HFC requires solving the linear prediction normal equations twice, once using the computed autocorrelation coefficients, and once using the modified autocorrelation coefficients. The first of the two solutions is required only to compute the normalized error V_p of the p-the order linear predictor (p=6 in our case). To reduce

the complexity of the first step, we used V_2 as an estimate of V_6, where V_2 can be computed explicitly in terms of R(0), R(1) and R(2) as given below:

$$V_2 = \left(1-[R(1)/R(0)]^2\right)\left(\left[1-\frac{R^2(1)-R(0)R(2)}{R^2(0)-R^2(1)}\right]^2\right) \qquad (39)$$

Using this second-order estimate, we reoptimized the parameter λ in Eq. (38) to be $\lambda = 0.035$, the previous choice being $\lambda = 0.05$. The original and the simplified HFC procedures were found to yield the same S/Q ratio and speech quality for the PP3-SQ3 coder. Therefore, we recommend the use of this simplified procedure in the real-time coder.

16.1.2 Pitch-Filter Stability Test

Since the optimized PP3-SQ3 coder uses 3-tap pitch prediction, it requires checking the stability of the pitch filter every frame and switching to 1-tap pitch prediction for frames for which instability is detected (see Section 8.2.2.1). The (exact) method of testing the pitch-filter stability requires 6M multiplies/frame, where M is the pitch period in number of samples: a nontrivial computation, especially for male speakers. To simplify the stability testing procedure, we considered an orthogonal linear transformation [35] of the three tap coefficients given below:

$$\left.\begin{aligned} T1 &= C1 + C2 + C3, \\ T2 &= C1 - 2C2 + C3, \\ T3 &= C1 - C3, \end{aligned}\right\} \qquad (40)$$

where we have denoted the tap coefficients C(M-1), C(M), and C(M+1) as C1, C2, and C3, for convenience. (This notation is also used in Appendix A.) Initially, we investigated a procedure that declared the pitch filter stable if the transformed coefficients satisfy the relations:

$$|T1| < 1, \ |T2| < 1, \ |T3| < 1; \qquad (41)$$

1-tap pitch prediction was used when the magnitude conditions (41) were *not* satisfied. Mathematically, the conditions (41) are neither necessary nor sufficient for pitch-filter stability. Experimentally, we found that as a detector of pitch filter instability, the above procedure yielded very high probability of detection (one error in 1200 frames) at the expense of a high false-alarm rate (declared instability for 20% of frames, while only 8% of frames had an unstable filter). Relative to the exact method, this simplified procedure yielded about 0.3 dB decrease in S/Q ratio and slight but audible speech quality degradation for 0% channel error, and it produced slightly more reverberant speech for 1% channel error because of the increased use of 1-tap prediction (see Section 12.2). Upon closer examination of the

false-alarm cases of the above procedure, we found that in each case T2 was in the range $-2<T2<1$. Therefore, we modified the "stability conditions" to be:

$$|T1| < 1, \; -2 < T2 < 1, \; |T3| < 1. \qquad (42)$$

This modified procedure yielded only 2 errors in the detection of instability out of the 1200 frames we considered. More important, this modified procedure yielded the same coder performance as the exact method both in the absence and in the presence of channel bit-errors. Therefore, the stability testing procedure involving Eqs. (40) and (42) is recommended for the real-time implementation.

16.1.3 Noise Shaping

The optimized PP3-SQ3 coder employs the pole-zero noise shaping method with a bandwidth parameter w of 800 Hz. Pole-zero noise shaping requires more computation and more coefficient and data memory than other types of noise shaping. The memory requirement is quite important for implementation on the MAP. The computational complexity of pole-zero noise shaping is roughly twice that of all-pole noise shaping and 2p (p=6 in our case) times that of 1-zero noise shaping. In an attempt to simplify the implementation, the all-pole and the one-zero noise

shaping methods were re-examined in systems otherwise identical to the optimized coder. For the all-pole method, w=200 Hz produced the best output speech quality. We then compared the speech produced by each of these two noise shaping methods against the speech produced by the pole-zero method. The speech for the all-pole method sounded more raspy and rough, and the 1-zero method produced noticeably more roughness (because of the use of preemphasis) and more background noise. The output speech obtained without noise shaping contained discrete distortions (e.g., clicks) and an increased level of roughness and background noise. Therefore, the complexity of the pole-zero method is worthwhile to keep in the real-time coder.

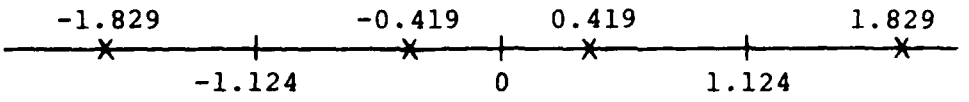
16.2 Refinements to the Coder

16.2.1 Laplacian versus Gaussian Quantizer

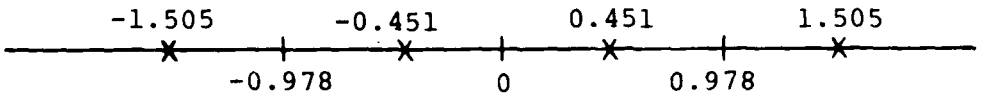
For the error-free channels, we had previously found that both Laplacian and Gaussian optimal nonuniform residual quantizers produced essentially the same perceived speech quality. We used the Laplacian quantizer in most of our simulations, because it produced about 0.5 dB higher S/Q ratio than did the Gaussian quantizer. However, when we repeated the same comparison for 1% channel error, we found that the Gaussian

quantizer produced a noticeable improvement in the speech quality over the Laplacian quantizer. The extent of reverberant quality and the loudness of discrete noises in the output speech were reduced with the use of the Gaussian quantizer. The observed difference in channel-error performance may be explained as follows. Using the decoded values for the two quantizers shown in Fig. 16 and assuming single bit-errors in the residual code, we can show that the mean-square decoding errors for the four levels in FBC (01,00,10,11) are 7.68, 1.35, 1.35, and 7.68 for the Laplacian case and 5.39, 0.96, 0.96, and 5.39 for the Gaussian case. Therefore, for each level, the Gaussian quantizer produces a lower mean-square decoding error than the Laplacian quantizer. This, therefore, explains the observed improvement produced by the Gaussian quantizer over the Laplacian quantizer.

Also, we observed that the benefit provided by the folded binary code was less in the Gaussian case than in the Laplacian case. This result can be explained by the larger width and hence the larger probability of occurrence of the inner levels for the Laplacian case than for the Gaussian case (see Fig. 16) and by the result presented in Section 12.7. The Gaussian quantizer was still judged to be better than the Laplacian quantizer when the two cases were compared, both using FBC. Therefore, we recommend the use of the Gaussian quantizer in the final coder design.



(a) Laplacian quantizer



(b) Gaussian quantizer

FIG. 16. Optimum Laplacian and Gaussian quantizers. The tick-marks are used to indicate quantizer input boundaries, and the symbols x are used to indicate quantizer output values.

16.2.2 Recomputing Parameter Quantization Tables

Having completed the coder design, we recomputed the statistics of each transmission parameter for the purpose of checking the ranges and step sizes used for the quantization. Using the 12-sentence high-quality data base, histograms were prepared for each parameter. Maximum and minimum parameter values (to be used in quantization) were then estimated by visual inspection of the histograms. As a result of the recomputed

statistics, we revised the quantization of log area ratios and delta gains. The revised quantization of delta gains is given below in Table 11, while the LAR quantization is included in the overall description of the optimized coder given in the next section.

Quantizer Input	Level	Quantizer Output
$-\infty$		
	1	-6.2
-3.6		
	2	-2.0
-0.5		
	3	1.0
2.2		
	4	3.5
∞		

TABLE 11. Revised nonuniform quantization of delta gains.

16.3 Optimized Coder Description

Before we describe the optimized coder, we discuss the choice of the frame size to be used in the real-time coder. Recall that the sampling rate of the real-time system is 6.621 kHz, while that used in the simulations is 6.67 kHz (Section 2.2). We found 32.625 ms to be the best choice of the frame size for the real-time system corresponding to the value of 33.75 ms

that we used in our simulations. Below, we summarize the details of the optimized, robust 16 kb/s coder. A detailed specification of the coder design is given in Appendix A.

A block diagram of the optimized coder is shown in Fig. 17. Table 12 provides information regarding the quantization and error protection of parameter data of the APC system. At the transmitter, the analog input speech is lowpass filtered at 3.2 kHz and sampled at 384/58 (or about 6.621) kHz. Referring to Fig. 17(a), the sampled speech s(t) is divided into frames of 216 samples (32.625 ms duration). Each frame of speech is preemphasized using the filter $(1 - 0.4z^{-1})$. The preemphasized speech $s'(t)$, before being encoded by the APC encoder, is processed, as explained below, to extract in order pitch predictor parameters, spectral predictor parameters, and segment gains of the quantizer. Extraction of the pitch predictor parameters consists of the following steps: computing the autocorrelation function of $s'(t)$ for lags 0-134, from an interval of 265 samples (216 from the current frame and 49 from the previous frame); determining the pitch value M as the peak of this function over lags 14-133; solving for the 3 pitch taps for the 3-tap filter and for the single tap for the 1-tap filter from the corresponding autocorrelation normal equations; checking for the stability of the 3-tap filter using Eqs.(40) and (42) and, if

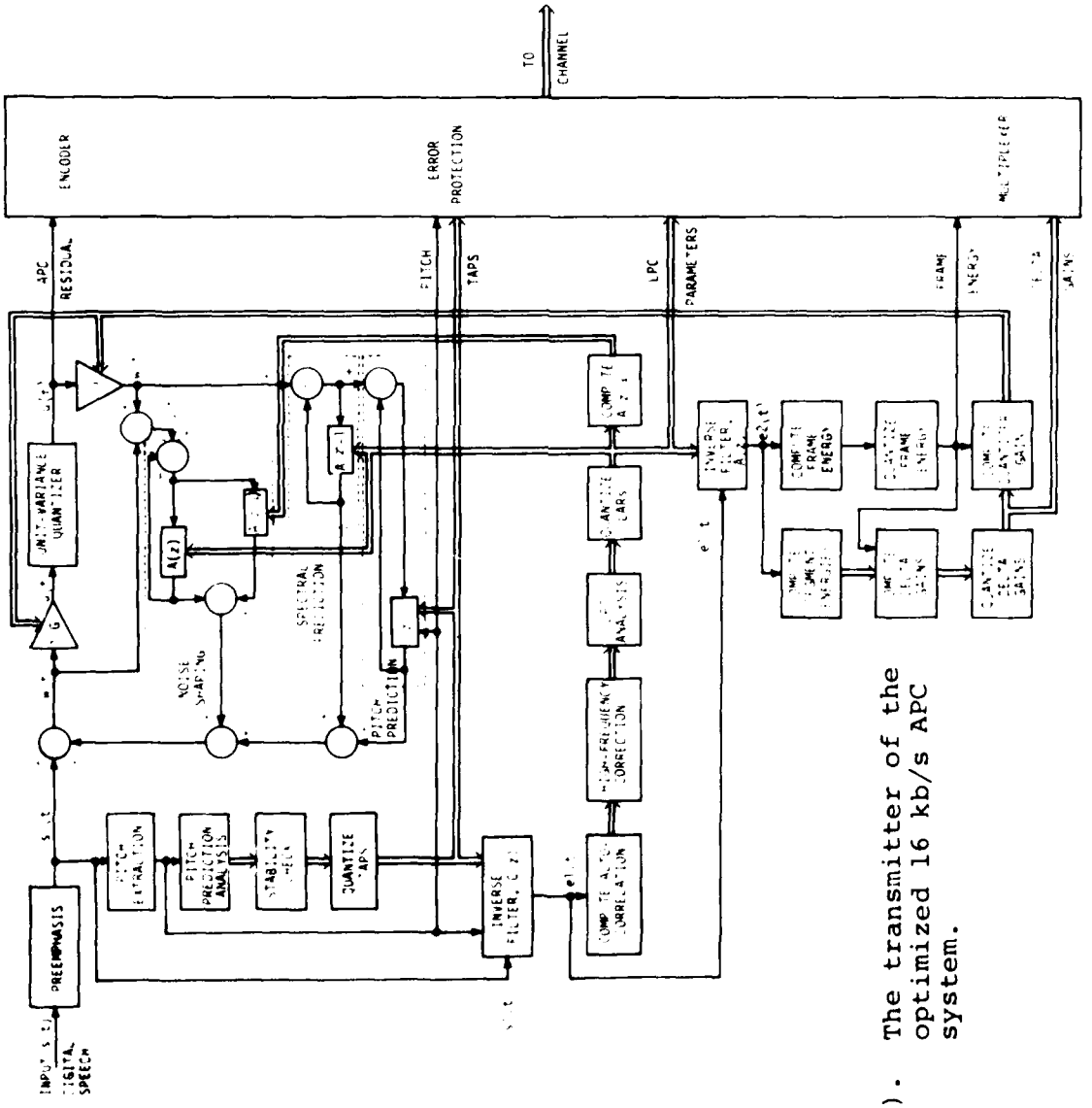


FIG. 17(a). The transmitter of the optimized 16 kb/s APC system.

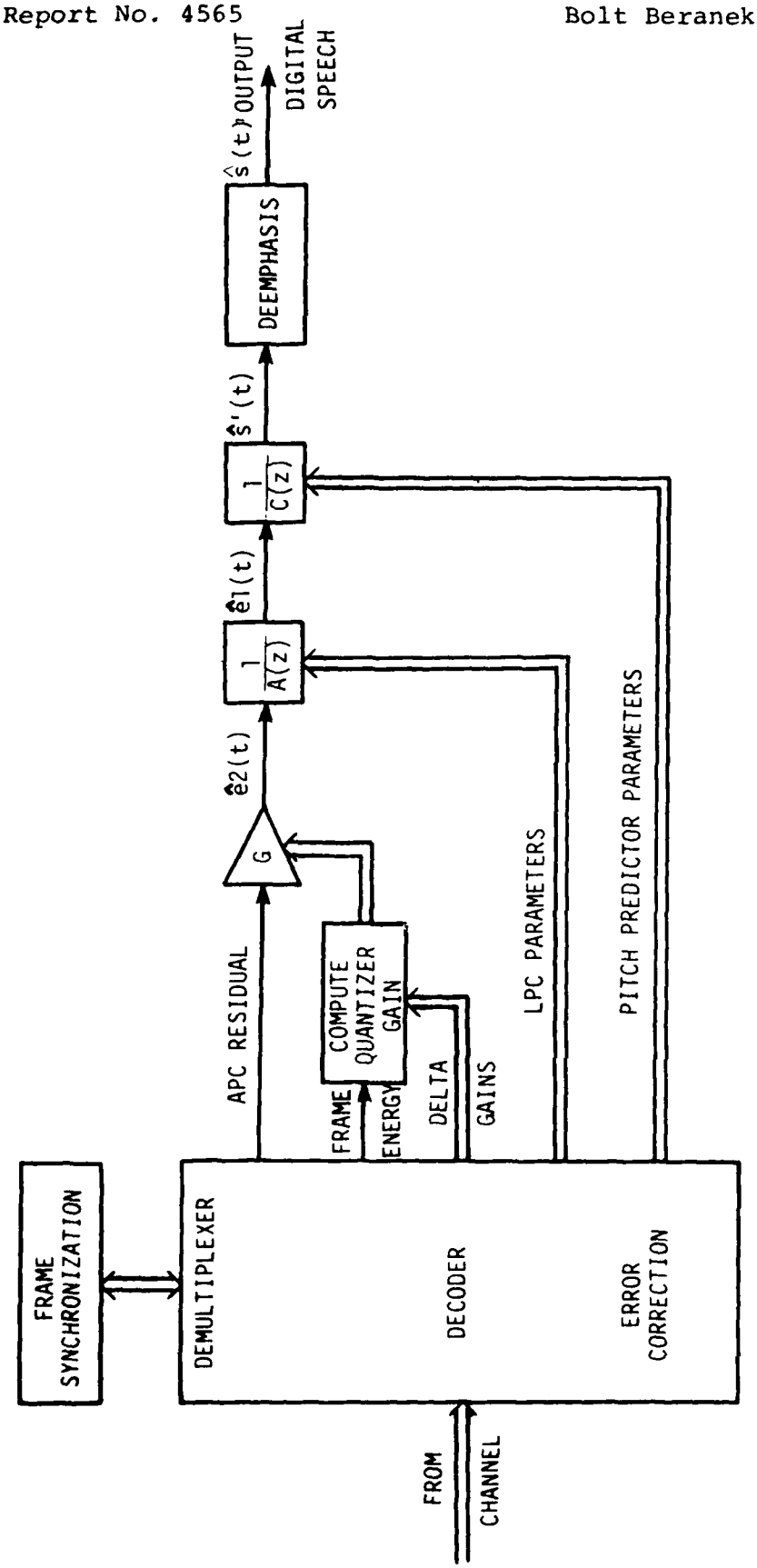


FIG. 17(b). The receiver of the optimized 16 kb/s APC coder.

Parameter		Min	Max	Step Size	# of Bits	# of Bits Protected
Pitch		14	133	No Quantization	7	7
Pitch Taps	C1	-0.549	0.427	0.122	3	2
	C2	-0.95	0.12	0.067	4	3
	C3	-0.549	0.427	0.122	3	2
Second residual energy (dB)		-10.0	46.0	0.875	6	6
Delta gains	1	See	Table	11	2	2
	2				2	2
	3				2	2
Log Area Ratios (dB)	1	-21.849	11.053	0.514	6	5
	2	- 8.789	13.711	0.703	5	4
	3	- 9.031	7.969	1.063	4	3
	4	- 6.094	8.906	0.938	4	2
	5	- 5.281	7.719	0.813	4	2
	6	- 3.741	9.559	0.831	4	2
Total bits per frame					56	44

TABLE 12. Quantization and error protection of parameter data for the optimized 16 kb/s APC system.

found unstable, replacing the computed 3-tap filter with another 3-tap filter with zero side taps and its center tap equal to the computed 1-tap coefficient; quantizing the 3 pitch taps; and inverse filtering the signal $s'(t)$ to produce the first residual $e1(t)$, using the pitch-inverse filter $C(z)$ with quantized coefficients. Extraction of spectral parameters consists of the following steps: computing the autocorrelation function of $e1(t)$ for lags 0-6; modifying the computed values of this autocorrelation function using Eqs.(38) and (39) and with λ=0.035; obtaining the reflection coefficients via autocorrelation LPC analysis; quantizing the reflection coefficients (via the log area ratio transformation); computing the coefficients of the numerator of the noise shaping filter, $A(z/\alpha)$ from the LPC predictor coefficients; and inverse filtering of the signal $e1(t)$ to produce the second residual $e2(t)$, using the spectral inverse filter $A(z)$. Extraction of the segment quantizer gains consists of the following steps: computing the energy of $e2(t)$ over the frame and over each of the three 72-sample segments in the frame; quantizing the frame energy; computing the three delta gains as the ratio of the segment energy and the quantized frame energy; quantizing the delta gains; and computing the segment quantizer gains from the quantized frame energy and the quantized delta gains. The quantized values of the various extracted parameters are used to

update the corresponding parameters of the APC encoder, which is set up as in the APC-PF configuration (see Fig. 4). The residual quantizer in the APC encoder is the optimal, 4-level, Gaussian, nonuniform quantizer (see Fig. 16(b)). The quantized parameter data, 3 pitch taps, 6 LARs, and frame energy and segment delta gains of the second residual, and the unquantized pitch are all binary encoded, error protected using 11 Hamming (7,4) codewords (see Table 12), multiplexed with one synchronization bit and 432 bits (2 bits/sample x 216 samples) of folded-binary encoded residual data, and transmitted over the channel.

At the receiver, shown in Fig. 17(b), the received data are demultiplexed, decoded, and error-corrected. The three segment quantizer gains are computed from the decoded frame energy and the delta gains. The decoded APC residual samples are multiplied by the corresponding segment quantizer gain and filtered first by the spectrum-synthesis filter $1/A(z)$ and then by the pitch-synthesis filter $1/C(z)$. The filtered output $\hat{s}'(t)$ is deemphasized using the filter $1/(1 - 0.4z^{-1})$ to produce the digital speech output $\hat{s}(t)$ (as an approximation to the original input $s(t)$). This digital output is passed through a D/A converter and an analog lowpass filter with its cutoff at 3.2 kHz to produce the analog output speech.

The COTR was supplied with an audio demonstration tape in

June 1980. The tape contained the recordings of the output speech obtained from the simulation of the above described 16 kb/s APC system. The recorded sections on the tape successfully demonstrated the performance of the robust coder, respectively, for high-quality input speech, in acoustic background noise, over a noisy channel in 1% bit-errors, and in tandem with the 2.4 kb/s LPC-10 coder. In each of these cases, the coder performance met the requirements stated in Chapter 1.

16.4 FORTRAN Simulation of the Optimized Coder

During the project, we developed a general software package to simulate the APC coder. It contained many features that aided us in program debugging and in the coder optimization and evaluation. This general software package was modified to produce a FORTRAN simulation of only the final optimized system.

A user's guide for this FORTRAN simulation is included with this report as Appendix B, and a listing of the FORTRAN source programs is contained in Appendix C. We have tested and verified that the FORTRAN simulation of the optimized coder produced synthesized speech identical to that produced by the general software package with the parameters set as in the optimized coder.

16.5 Results of Tests on the Real-Time Coder

We tested the real-time 16 kb/s coder on the MAP-300, using input from several tapes containing speech from a number of males and females, for informal evaluation of the coder performance for different speech materials and different speakers. Except for two problems mentioned below, the coder was found to produce high quality speech output. First, the output speech for one low-pitched male talker (with an average pitch of 95 Hz) contained audible roughness. Second, the coder produced audible background noise at the output for some female talkers.

To investigate the causes of these problems, we performed several tests on the real-time coder and on the FORTRAN simulation. First, using the RT-11 debugging program (FDT) on our PDP-11, the values of three of the coder parameters, which are specified in DATA statements, were varied about their nominal (previously optimized) values. The three parameters are: preemphasis constant β, noise shaping bandwidth parameter w, and high-frequency correction coefficient λ. After each parameter change, we listened to the output of the real-time coder, with its input speech from a tape. For each of the parameters, we concluded that the nominal value produced the best overall speech quality.

Second, using the CSPI-supplied program MPLOOK, we made changes to the coding and decoding tables of the 4-level residual quantizer. Two types of changes were investigated: 1) Each element of the coding and decoding tables was multiplied by a constant (called quantizer load factor) to investigate the tradeoff between clipping and granular quantization errors; and 2) different types of unit-variance quantizers were employed. For the first item, we used values of 0.8, 1.0 (nominal value), and 1.2 as load factors. For the second item, we compared Laplacian and gamma quantizers with the nominal Gaussian quantizer. This investigation of changes to the quantizer also resulted in no perceivable improvement in the overall speech quality of the coder.

Since the two types of testing, described above, on the real-time coder did not uncover the observed speech-quality problems, we decided to pursue the subsequent work using our earlier versions of the FORTRAN simulation of the APC coder. We chose two specific sentences that suffered the greatest quality degradation and digitized them at the 6.67 kHz sampling rate. We processed one of these sentences (spoken by a low-pitched male talker) using the simulation program, without quantization of any parameters (i.e., with only the residual being quantized). The output speech was found to be nearly identical to the output of

the real-time coder. We then investigated the following changes to the coder (one change at a time), but we observed no significant improvement in the coder performance: 1) Pitch filter stability check was not used; 2) analysis frame size used for pitch computation was varied between 35 and 45 ms; and 3) LPC order used in spectral prediction was increased from 6 to 10 poles.

In a subsequent set of tests, we found that each of the following changes did produce a significant increase in speech quality: (1) variable-rate entropy coding, with an average entropy of 2 bits/sample; (2) increase from 3 (optimized value) to 10 in the number of segments used for segmented quantization; and (3) use of pitch-adaptive quantization. We did not use variable-to-fixed rate conversion in (1), and we did not readjust the bit allocation to limit the data rate to 16 kb/s in (2). For the low-pitched male speaker, the entropy coding and 10-segment schemes each produced slightly higher speech quality than the pitch-adaptive scheme. For the second sentence from a female speaker, increasing the number of segments to 10 did not improve the speech quality. Also, we found that the 5-segment scheme produced about the same overall speech quality as the 3-segment scheme, even for the low-pitched male speaker.

Based on these experimental investigations, we offer the

following conclusions. The observed speech quality degradations were caused by the relatively large dynamic range of the input to the residual quantizer. Both entropy coding and pitch-adaptive quantization methods represent effective ways of dealing with the problem. However, the performance of the entropy coding method under channel errors is substantially worse than the performance produced by the optimized coder (see Section 12.6). As for the pitch-adaptive method, its implementation on the MAP is extremely difficult, as we reported in Chapter 11. Increasing the number of segments from 3 to 10 prevents only the roughness problem observed for low-pitched males. Further, such a change would involve a reoptimization of the coder and may perhaps lead to a less robust channel-error performance than our original optimized coder. All things considered, we believe that the coder design implemented on the MAP is still the most robust coder meeting the design requirements given in Chapter 1. The test results reported in this section have shown that for some speakers, the coder output speech degrades perceivably relative to the input speech.

17. SUMMARY AND MAJOR CONTRIBUTIONS

In summary, we have investigated and compared several methods, some already existing ones and some new ones developed in this work, for coding the residual signal and for shaping the spectrum of the quantization noise, in the course of optimizing the APC system to meet the specific needs of this project. As part of this work, we have also optimized the values of various parameters as well as the bit a location for those parameters that are transmitted to the receiver, to produce the best output speech quality at a synchronous data rate of 16 kb/s and for an input-speech sampling rate of 6.67 kHz. For the noisy channel application, we have considered in the optimization study the tradeoff between the voice data rate and the error-protection rate and the allocation of the error protection bits among individual transmission parameters.

As a result of this work, we have developed two best 16 kb/s APC systems, one for use over perfect or noiseless channels and the other for noisy channel applications involving as much as 1% bit-errors. For an error-free transmission, the best system uses 8-pole spectral prediction, 3-tap pitch prediction, entropy coding with a large number of quantizer levels (43 levels used in our tests), and pole-zero noise shaping. For operation over

noisy channels, the most robust system uses 6-pole spectral prediction, 3-tap pitch prediction, 3-segment segmented quantization with a 4-level nonuniform Gaussian quantizer, and pole-zero noise shaping; allocates to error protection of parameters slightly over 6% of the total transmission bit rate (or about 37% of the bit rate used for parameter transmission); and encodes the quantized residual samples with the folded binary code. Informal listening tests have shown that the second system satisfies all the design requirements of this project: speech-quality requirements for high-quality speech inputs and for acoustic background noise environments, robustness requirement in channel bit-errors of 1%, and speech-intelligibility requirement for tandem operation with a 2.4 kb/s LPC-10 coder. We have made specific suggestions for improving the speech quality of the APC-LPC tandem. Quite impressively, the robust coder produces only a slight speech quality degradation as the channel bit-error rate is increased from 0% to 1%.

In this work, in addition to designing a robust APC coder that meets the requirements of this project, as mentioned above, we have made a number of significant contributions, which when put together represent, in our view, an advance in the state of the art in adaptive predictive coding of speech. The specific contributions of this work are stated below:

1. Demonstration of the important role played by the sequencing of spectral and pitch predictors.

2. Establishment of performance equivalence conditions for the several configurations of the APC system. (Any violation of these conditions has been found to yield a significant performance degradation.)

3. Demonstration of the effects of, and development of a successful remedy for, the instability problem of multi-tap pitch prediction.

4. Identification of excessive quantization-noise problems as the limit-cycle behavior of the quantizer output, interpretation of the causes of the limit cycles in terms of the feedback gain of the APC loop, and comprehensive solution of the limit-cycle problem by reducing the feedback gain.

5. Demonstration of the dual benefits of noise shaping: suppression of quantization-noise perception and reduction of feedback gain, and of how the role of noise shaping is affected by other system components (e.g., preemphasis).

6. Development of new methods for the coding of APC residual: multi-tap pitch prediction and segmented quantization; pitch-adaptive coding with multi-tap pitch prediction and pitch-synchronous segmented quantization and using variable number of bits/sample over segments; and segmented quantization with multi-tap pitch prediction and using a variable number of bits/sample over segments.

7. Demonstration of the importance of (multi-tap) pitch prediction for significantly improving the coder performance both over noiseless channels and over noisy channels. That a robust APC coder design must include pitch prediction has been vividly demonstrated in one of our experiments comparing three 16 kb/s entropy-coded (0-tap, 1-tap and 3-tap) systems operating in 1% channel error.

REFERENCES

1. Proposal for Speech Algorithm Optimization at 16 kb/s, BBN Proposal No. P79-ISD-42, Bolt Beranek and Newman Inc., February 1979, Submitted to Defense Communications Agency.

2. J.L. Flanagan, "Opportunities and Issues in Digitized Voice," Proceedings EASCON '78, EASCON, Washington, D.C., September 1978, pp. 709-712.

3. R. Viswanathan, J. Wolf, L. Cosell, K. Field, A. Higgins, and W. Russell, "Design and Real-Time Implementation of a Baseband LPC Coder for Speech Transmission Over 9600 Bps Noisy Channels - Final Report," BBN Report No. 4327, Bolt Beranek and Newman Inc., February 1980.

4. R. Viswanathan, A. Higgins, W. Russell and J. Makhoul, "Baseband LPC Coders for Speech Transmission over 9.6 kb/s Noisy Channels," IEEE International Conf. Acoustics, Speech and Signal Processing, IEEE, Denver, CO, April 1980, pp. 348-351.

5. J. Makhoul and M. Berouti, "Adaptive Noise Spectral Shaping and Entropy Coding in Predictive Coding of Speech," IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-27, February 1979, pp. 63-73.

6. R. Zelinski and P. Noll, "Adaptive Transform Coding of Speech Signals," IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-25, August 1977, pp. 299-309.

7. B.S. Atal and M.R. Schroeder, "Adaptive Predictive Coding of Speech Signals," Bell Syst. Tech. J., Vol. 49, October 1970, pp. 1973-1986.

8. N.S. Jayant, "Digital Coding of Speech Waveforms: PCM, DPCM, and DM Quantizers," Proc. IEEE, Vol. 62, May 1974, pp. 611-632.

9. P. Noll, "A Comparative Study of Various Quantization Schemes for Speech Encoding," Bell Syst. Tech. J., Vol. 54, November 1975, pp. 1597-1614.

10. A.J. Goldberg, R. L. Freudberg and R. S. Cheung, "High Quality 16 kb/s Voice Transmission," IEEE International Conf. Acoustics, Speech and Signal Processing, IEEE, Philadelphia, PA, April 1976, pp. 244-246.

11. B.S. Atal and M.R. Schroeder, "Predictive Coding of Speech Signals and Subjective Error Criteria," IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-27, June 1979, pp. 247-254.

12. D.L. Cohn and J.L. Melsa, "The Residual Encoder - An Improved ADPCM System for Speech Digitization," IEEE Trans. Communications, Vol. COM-23, September 1975, pp. 935-941.

13. J.Makhoul, "Linear Prediction: A Tutorial Review," Proc. IEEE, Vol. 63, April 1975, pp. 561-580.

14. J. Max, "Quantizing for Minimum Distortion," IRE Trans. Info. Theory, Vol. IT-6, March 1960, pp. 7-12.

15. M.D. Paez and T.H. Glisson, "Minimum Mean-Squared-Error Quantization in Speech PCM and DPCM Systems," IEEE Trans. Commun. Tech., Vol. COM-20, April 1972, pp. 224-230.

16. P. Noll, "Adaptive Quantizing in Speech Coding Systems," Proc. 1974 IEEE Zurich Seminar on Digital Commun., IEEE, March 1974, pp. B3(1)-B3(6). (Reprinted in the IEEE Press Book Waveform Quantization and Coding, N.S.Jayant, (Ed.), 1976, pp.204-209.)

17. B.J. McDermott, C. Scagliola and D. Goodman, "Perceptual and Objective Evaluation of Speech Processed by Adaptive Differential PCM," Bell Syst. Tech. J., Vol. 57, May-June 1978, pp. 1597-1618.

18. J.M. Tribolet, P. Noll, B.J. McDermott and R.E. Crochiere, "A Study of Complexity and Quality of Speech Waveform Coders," Proc. 1978 IEEE International Conf. Acoustics, Speech and Signal Processing, IEEE, Tulsa, OK, April 1978, pp. 586-590.

19. C.C. Cutler, "Transmission Systems Employing Quantization," U.S. Patent 2,927,962, 1960.

20. E.G. Kimme and F.F Kuo, "Synthesis of Optimal Filters for a Feedback Quantization System," IEEE Trans. Circuit Theory, Vol. CT-10, 1963, pp. 403-413.

21. R.C. Brainard and J. C. Candy, "Direct-Feedback Coders: Design and Performance with Television Signals," Proc. IEEE, Vol. 57, May 1969, pp. 776-786.

22. P. Noll, "On Predictive Quantizing Schemes," Bell Syst. Tech. J., Vol. 57, May-June 1978, pp. 1499-1532.

23. R. Viswanathan and J. Makhoul, "Quantization Properties of Transmission Parameters in Linear Predictive Systems," IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-23, June 1975, pp. 309-321.

24. R. Viswanathan, J. Makhoul and A.W.F. Huggins, "Speech Compression and Evaluation," BBN Report No. 3794, Bolt Beranek and Newman Inc., April 1978.

25. E. Blackman, R. Viswanathan, W. Russell and J. Makhoul, "Narrowband LPC Speech Transmission over Noisy Channels," IEEE International Conf. Acoustics, Speech and Signal Processing, IEEE, Washington, D.C., April 1979, pp. 60-63.

26. J. Makhoul, "Stable and Efficient Lattice Methods for Linear Prediction," IEEE Trans. Acoustics, Speech and Signal Processing, Vol. ASSP-25, October 1977, pp. 423-428.

27. T.E. Tremain, J.W. Fussell, R.A. Dean, B.M. Abzug, M.D. Cowing and P.W. Boudra, Jr., "Implementation of Two Real-Time Narrowband Speech Algorithms," Proc. EASCON '78, EASCON, Washington, D.C., September 1978, pp. 698-708.

28. D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," Proc. IRE, Vol. 40, September 1952, pp. 1098-1101.

29. M. Berouti, J. Makhoul, and M. Krasner, "Efficient Encoding and Decoding of Speech," BBN Report No. 4567, Bolt Beranek and Newman Inc., Dec. 1980.

30. R. Hamming, Introduction to Applied Numerical Analysis, McGraw-Hill Book Company, 1971.

31. F. Itakura, "Research on Low Bit Rate Speech Coding at the Electrical Communication Laboratory, NTT," Presented at the 96th Meeting of the Acoust. Soc. Amer., Honolulu, Hawaii, November 27-December 1, 1978.

32. N.S. Jayant, "Average- and Median-Based Smoothing Techniques for Improving Digital Speech Quality in the Presence of Transmission Errors," IEEE Trans. Communications, Vol. COM-24, September 1976, pp. 1043-1045.

33. M. Berouti, R. Schwartz and J. Makhoul, "Enhancement of Speech Corrupted by Acoustic Noise," 1979 IEEE International Conf. Acoustics, Speech and Signal Processing, IEEE, Washington, D.C., April 1979, pp. 208-211.

34. T.P. Barnwell, R.W. Schafer, and A.M. Bush, "Evaluation of LPC/CVSD Tandem Connections," , IEEE International Conf. Acoustics, Speech and Signal Processing, Tulsa, OK, Apr. 1978, pp. 326-329.

35. B.S. Atal, Personal Communication with R. Viswanathan.

APPENDIX A

SPECIFICATION OF THE OPTIMIZED 16 KB/S APC ALGORITHM

TABLE OF CONTENTS

Table of Contents (Cont.)

1. SOME GENERAL ITEMS

1.1 Speech signal sampling rate = 384/58 KHz (~6.621 KHz)

1.2 Frame size = 32.625 ms or 216 samples

1.3 Spectral predictor order = 6

1.4 Pitch predictor order = 3

1.5 Parameter Coding

Coding and decoding tables (Tables 1-5) are given at the end of this appendix. Each of the tables has three columns, X(J),J,R(J), where

X(J) = quantization boundary
J = code or level
R(J) = decoded or quantized parameter value.

When a parameter has a value A, which satisfies $X(J) \leq A < X(J+1)$, it is coded as J and decoded as R(J).

1.6 Data rate = 16 kb/s or 522 bits/frame

Item	Bits/frame
Parameter data	56
Protection	33
Residual samples (216 X 2)	432
Sync	1
Total	522

1.7 Bit Allocation for Quantization and Protection

Parameter		Bits (Total)	Most Significant Bits Protected
Reflection Coefficients	IK(1)	6	5
	IK(2)	5	4
	IK(3)	4	3
	IK(4)	4	2
	IK(5)	4	2
	IK(6)	4	2
Gain	IG	6	6
Pitch	IM	7	7
Delta Gains	IDG(1)	2	2
	IDG(2)	2	2
	IDG(3)	2	2
Pitch Taps	IC(1)	3	2
	IC(2)	4	3
	IC(3)	3	2
Total		56	44

The 44 bits are protected using 11 Hamming (7,4) codewords. Error protection and correction are done as in our 9.6 kb/s BBC coder [1], and therefore these items are not discussed below.

2. TRANSMITTER

A block diagram of the transmitter is given in Figure 1. In this section, we specify each of the various transmitter components.

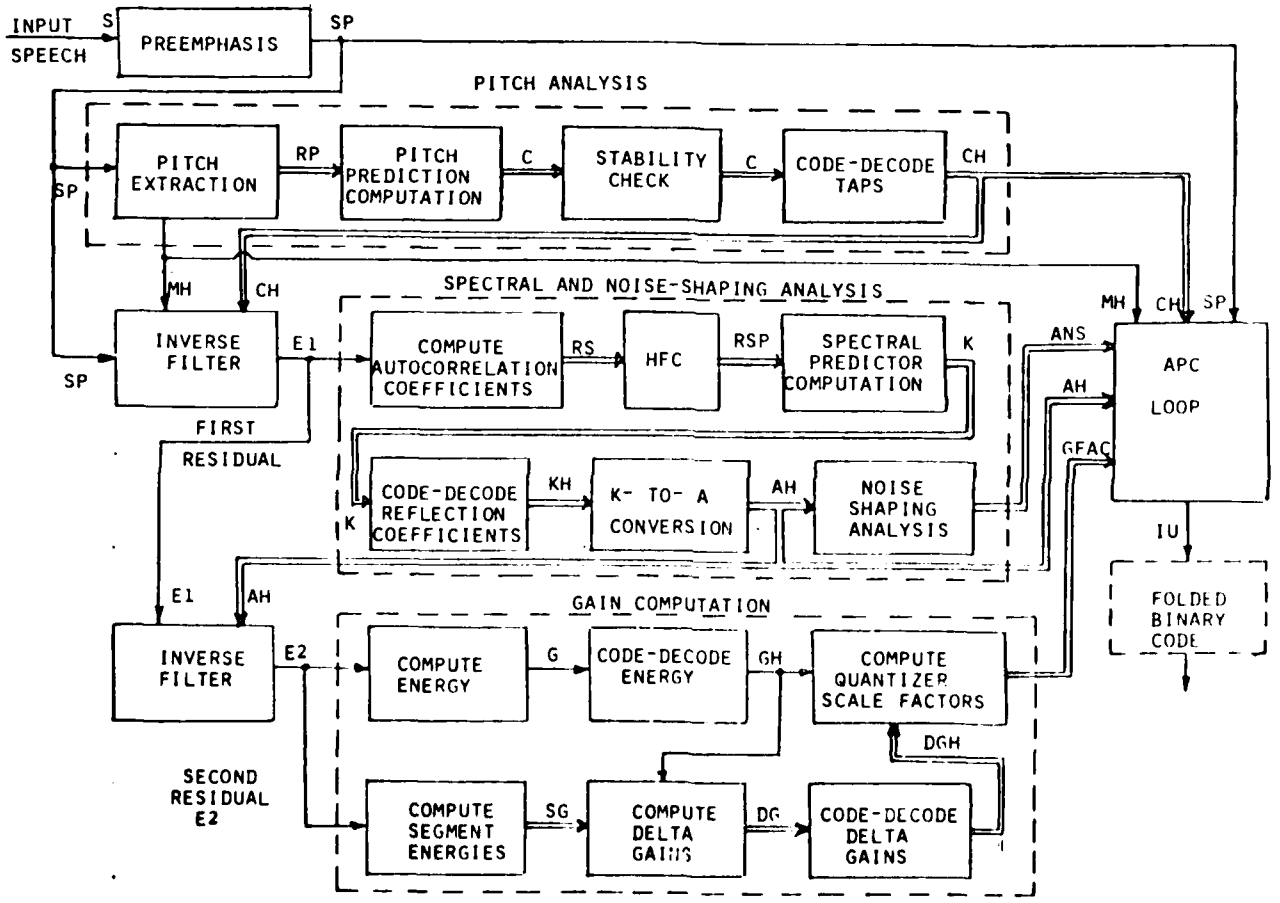


Figure 1. Block diagram of the transmitter of the APC coder

2.1 Preemphasis

$$SP(n) = S(n) - ALPHA * S(n-1), \quad 1 \leq n \leq 216,$$

where

S = input speech samples (216 samples of the present frame; last one sample of the past frame)
ALPHA = constant = 0.4
SP = output samples (216 total)

Save the last input sample of the present frame as initial condition of the next frame.

2.2 Pitch Analysis

Pitch analysis consists of the following steps, as shown in Figure 2: pitch extraction, pitch prediction computation, stability check, and coding and decoding of pitch and pitch taps. The symbols given in Figure 2 denote the various quantities as listed in the next page.

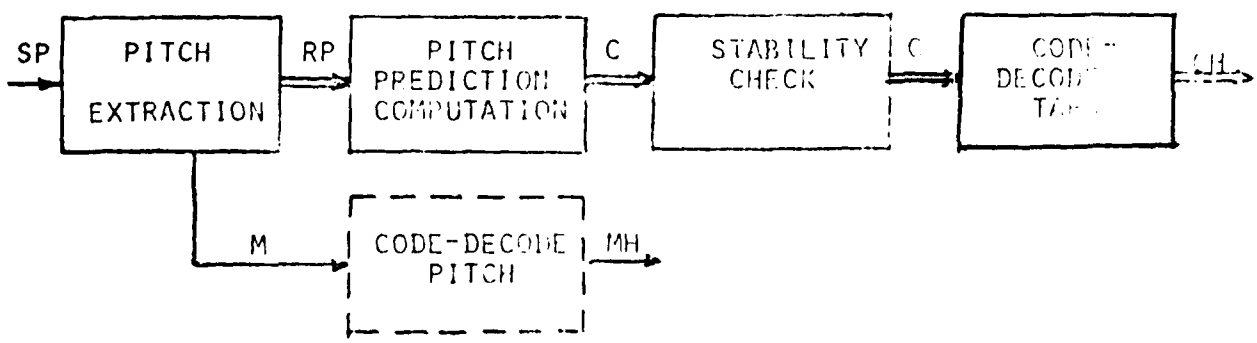


Figure 2. Block diagram for the pitch analysis

SP = preemphasized speech
RP = autocorrelation coefficients of SP
M = pitch period in number of samples
C = pitch predictor taps: C1,C2,C3
CH = quantized taps
MH = quantized pitch

2.2.1 Pitch Extraction

Pitch extraction uses a frame of 265 preemphasized speech samples (~40 ms): 216 samples of the present frame and 49 samples from the past frame. Pitch extraction consists of the following sequence of operations: remove DC, hamming window, compute autocorrelation coefficients, RP, and compute pitch, M.

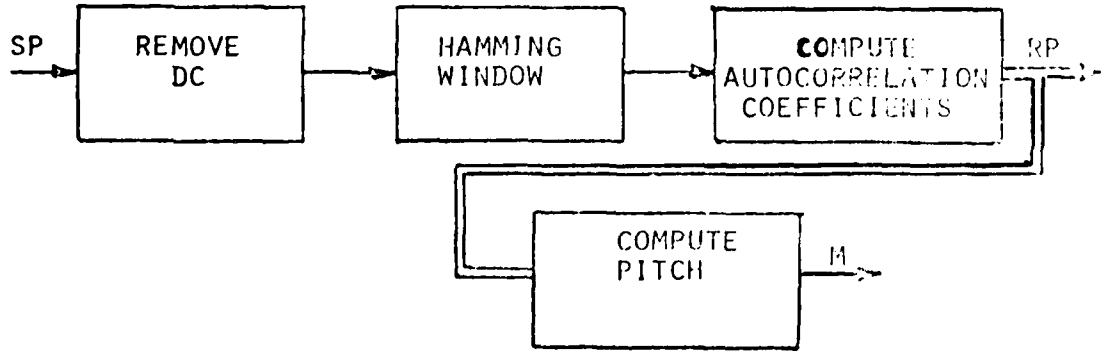


Figure 3. Block diagram for the pitch extraction

2.2.1.1 Remove DC

$y(n) = x(n) - DC, \quad 1 \leq n \leq N,$

where

x = input preemphasized samples SP (265 total)
y = output samples (265 total)
DC = 1/N*SUM[x(i),1≤i≤N],N=265

2.2.1.2 Hamming Window

$$y(n) = x(n)\left\{ ALPHA-BETA*\cos[2\pi(n-1)/(N-1)]\right\} ,1\leq n\leq N,N=265,$$

where

x = input preemphasized and DC-removed samples, (265 total)
y = output samples (265 total)
ALPHA = 0.54
BETA = 1.0 - ALPHA = 0.46

2.2.1.3 Compute Autocorrelation Coefficients

Direct Method

$$RP(m) = SUM[x(n)*x(n+m),1\leq n\leq N-m],0\leq m\leq MX,N=265,$$

where

x = Hamming-windowed input samples (265 total)
RP(m) = autocorrelation coefficient of lag m
MX = maximum lag = 134

FFT Method

(a) pad with zeros

$$x(n) = 0.0 \ ,266\leq n\leq 512$$

(b) Compute 512-point FFT of x(n)

$$X(k) = FFT(x(n)), \ 1\leq k\leq 512$$

(c) Compute power spectrum of x(n)

$$|X(k)|^2 = [X_R(k)]^2 + [X_I(k)]^2$$

where

$X_R(k)$ = real part of X(k)
$X_I(k)$ = imaginary part of X(k)

(d) Compute 512-point inverse FFT of $|X(k)|^2$

$V(m) = FFT^{-1}[|X(k)|^2]$, $1 \leq m \leq 512$

(e) Autocorrelations are defined as:

$RP(m), = V(m+1), \quad 0 \leq m \leq MX$

NOTE: It is possible to reduce the computation, as follows:

Since input sequence x(n) is real,

$X_R(k)$ is even (i.e. $X_R(k) = X_R(512-k)$), and
$X_I(k)$ is odd (i.e. $X_I(k) = -X_I(512-k)$)

Therefore, in step (b) compute the lower half of FFT

$X(k)$, $1 \leq k \leq 257$,

and in step (c) compute the lower half of the power spectrum

$|X(k)|^2$, $1 \leq k \leq 257$.

Then, fill $|X(k)|^2$ array from k = 258 to 512 as:

$|X(k)|^2 = |X(512-k+2)|^2$, $258 \leq k \leq 512$.

Compute steps (d) and (e) as above.

2.2.1.4 Compute Pitch

Search the autocorrelation function RP(m) for a maximum between the range of m=14 to m=133. Pitch M is computed as the lag, m, at which the autocorrelation coefficient, RP(m), is maximum.

2.2.2 Pitch Prediction Computation

Compute the 3-tap and the 1-tap pitch predictor coefficients. (The 1-tap coefficient is used if the stability check of the 3-tap filter fails).

(a) Compute the 3-tap coefficients from the normal equations:

$$\begin{bmatrix} RP(0) & RP(1) & RP(2) \\ RP(1) & RP(0) & RP(1) \\ RP(2) & RP(1) & RP(0) \end{bmatrix} * \begin{bmatrix} C1 \\ C2 \\ C3 \end{bmatrix} = - \begin{bmatrix} RP(MH-1) \\ RP(MH) \\ RP(MH+1) \end{bmatrix}$$

where

RP = autocorrelation coefficients (6 total)
C1,C2,C3 = pitch predictor coefficients
MH = quantized pitch period (See Section 2.2.5 for pitch quantization)

The solution for the above normal equations may be obtained by the Levinson recursion or from expressions derived by solving the 3 equations. Note that the right-hand-side vector in the above normal equations does not have the elements RP(1), RP(2) and RP(3); this means that the recursive solution used in the standard autocorrelation method cannot be employed here.

(b) Compute the 1-tap coefficient, C2P, as:

C2P = -RP(MH)/RP(0)

2.2.3 Stability Check

Transform pitch predictor coefficients (C1,C2,C3) as:

T1 = C1 + C2 + C3

T2 = C1 - 2*C2 + C3

T3 = C1 - C3

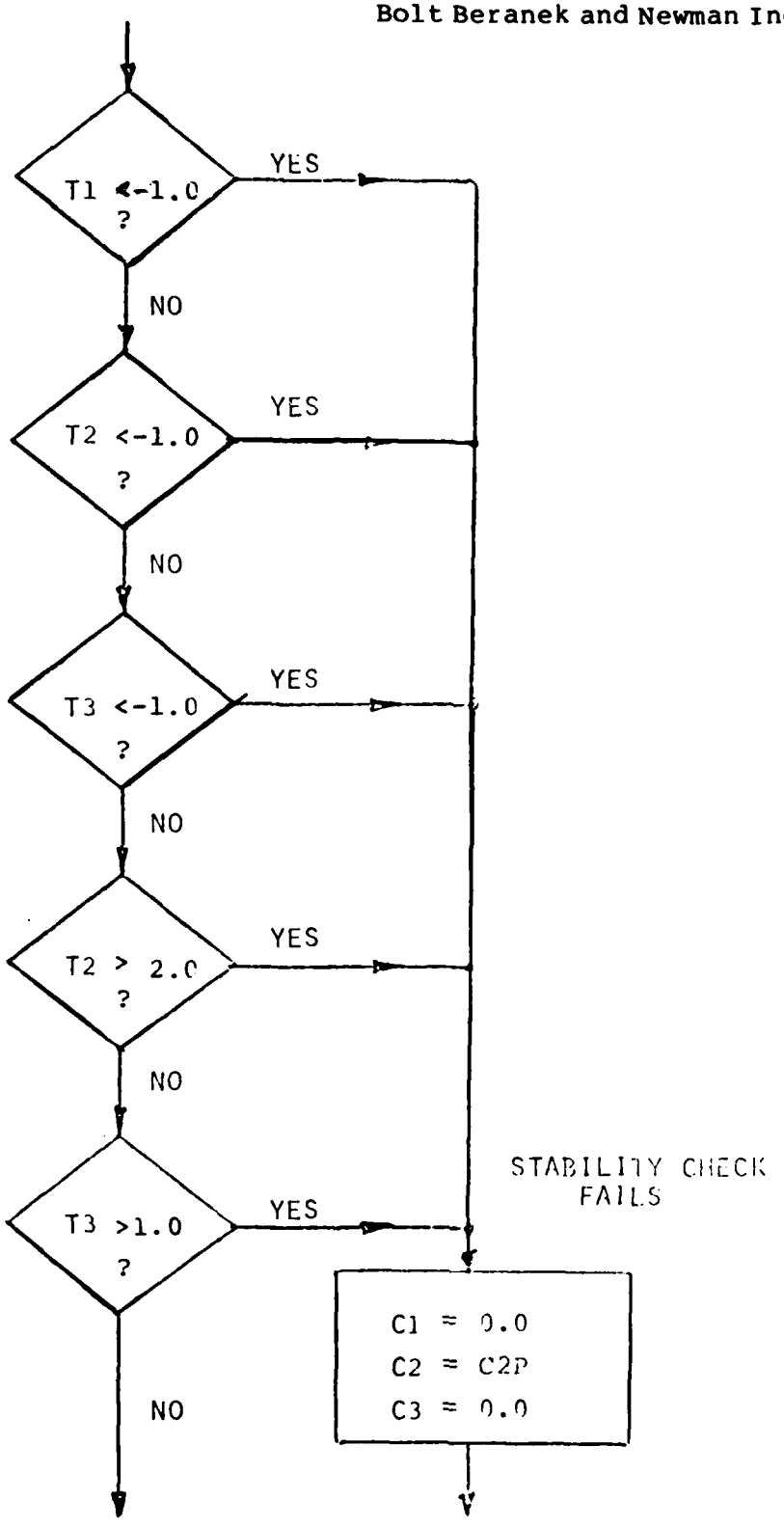


Figure 4. Flowchart for stability check

Check the range of transformed parameters (T1,T2,T3) as shown in Figure 4. Whenever the stability check fails, use the 3-tap predictor C1=0,C2=C2P, and C3=0, which is actually the 1-tap predictor.

2.2.4 Code-Decode Taps

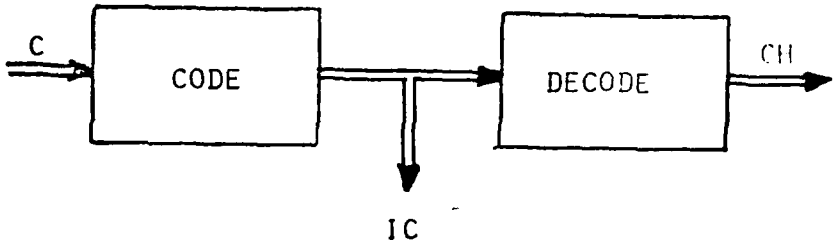


Figure 5. Encoding and decoding of the pitch prediction taps

where

C = pitch predictor coefficients: C1,C2,C3
IC = transmitted codes
CH = quantized values, C1H,C2H,C3H

The coefficients C1,C2, and C3 are coded using 3,4, and 3 bits, respectively. The coding and decoding tables for taps are given in Table 1.

2.2.5 Code-Decode Pitch

Since the pitch period M takes integer values in the range 14-133 (a total of 120 values), it is coded directly in 7 bits, as:

IM = M-14

Decoded value MH is given by:

MH = IM+14

Thus, pitch is quantized without error i.e., MH = M

2.3 Inverse Filter to Obtain First Residual

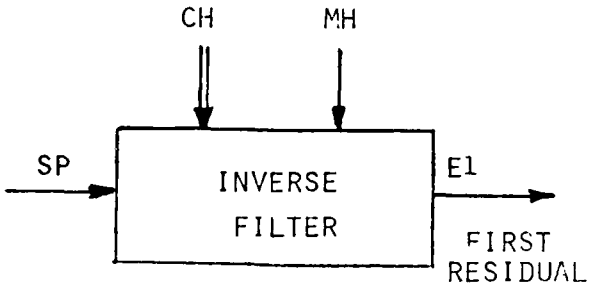


Figure 6. Inverse filter for first residual

E1(n) = SP(n) + C1H*SP(n-MH+1) + C2H*SP(n-MH)+C3H*SP(n-MH-1), where

SP = input preemphasized speech samples (216 samples of the present frame and up to 134 samples of the past frame)
CH = quantized pitch predictor coefficients: C1H.C2H,C3H
MH = quantized pitch
E1 = output samples of the first residual (216 total)

2.4 Spectral and Noise-Shaping Analysis

Spectral or linear prediction analysis consists of the following sequential steps: compute autocorrelation coefficients RS, high-frequency correction (HFC), spectral predictor computation, code and decode reflection coefficients, and convert quantized reflection coefficients to predictor coefficients (KH to AH). Noise-shaping analysis involves computing the predictor coefficients ANS from the coefficients AH. In Figure 7, we have used the following terminology:

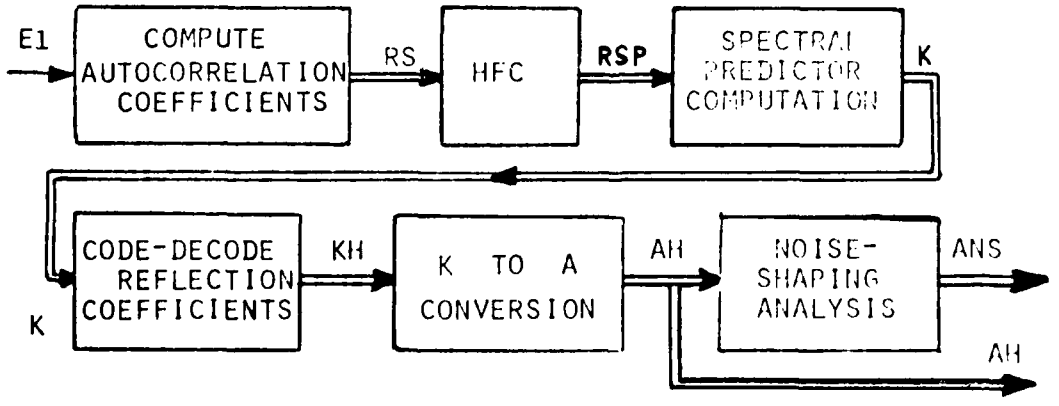


Figure 7. Block diagram of spectral and noise shaping analysis

El = first residual samples
RS = autocorrelation coefficients
RSP = high-frequency corrected autocorellation coefficients
K = reflection coefficients
KH = quantized reflection coefficients
AH = predictor coefficients
ANS = noise shaping filter coefficients

2.4.1 Compute Autocorrelation Coefficients

First, Hamming-window 216 samples of the first residual, El (see Section 2.2.1.2); then, compute the 7 autocorrelation coefficients RS(0), RS(1),...RS(6), using the direct method, as in 9.6 kb/s BBC coder [1].

2.4.2 High-Frequency Correction (HFC)

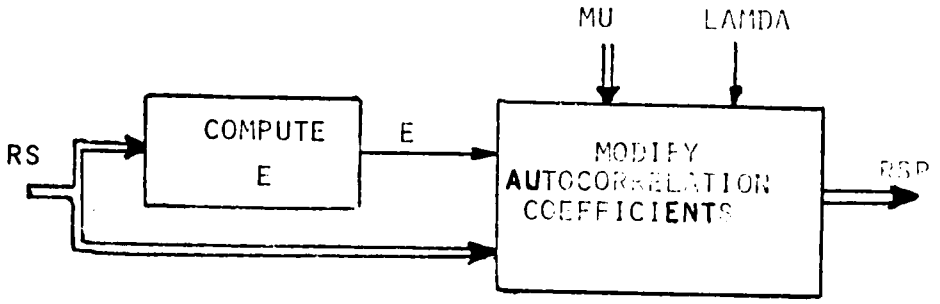
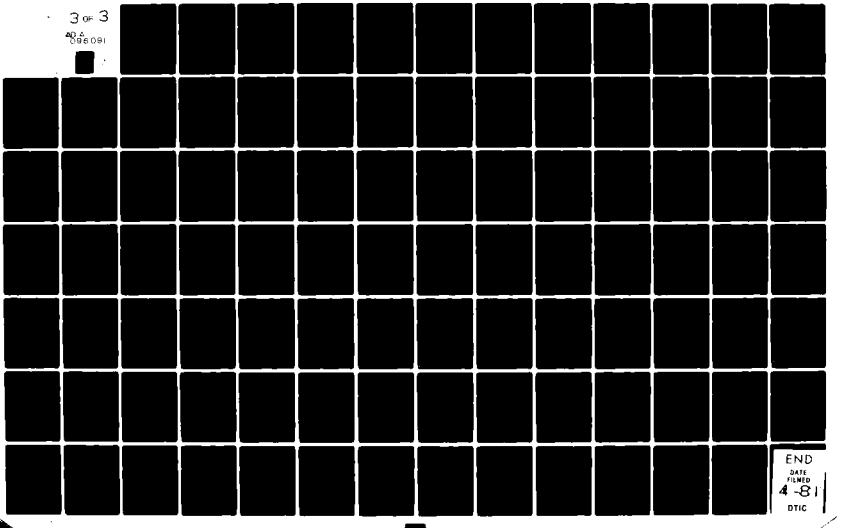


Figure 8. Block diagram of high-frequency correction.

AD-A096 091 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2.1
DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR S--ETC(U)
DEC 80 R VISWANATHAN, W RUSSELL, A HIGGINS DCA100-79-C-0037

UNCLASSIFIED NL

3 OF 3
AD A
096091



END
DATE FILMED
4-81
DTIC

2.4.2.1 Compute minimum mean-squared prediction error, E

$$E = RS(0) * \left[1.0 - \left(\frac{RS(1)}{RS(0)}\right)^2\right] * \left[1.0 - \left(\frac{RS(1)^2 - RS(0)*RS(2)}{RS(0)^2 - RS(1)^2}\right)^2\right]$$

where

RS = autocorrelation coefficients

2.4.2.2 Modify autocorrelation coefficients

$RSP(n) = RS(n) + LAMBDA*E*MU(n)$, $0 \leq n \leq 2$,

$RSP(n) = RS(n)$, $3 \leq n \leq 6$,

where

RS = autocorrelation coefficients (7 total)
LAMBDA = 0.035
MU(0) = +0.375
MU(1) = -0.25
MU(2) = +0.0625
RSP = output autocorrelation coefficients (7 total)

2.4.3 Spectral Predictor Computation

Use the standard routine employed in the 9.6 kb/s BBC coder [1]. The input and output quantities are:

RSP = input autocorrelation coefficients (7 total)
K = output reflection coefficients: K(1),K(2),...K(6)

2.4.4 Code-Decode Reflection Coefficients

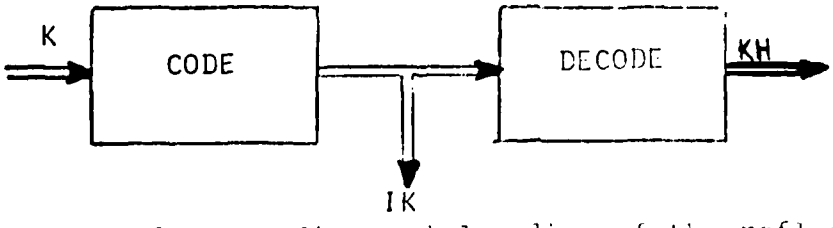


Figure 9. Encoding and decoding of the reflection coefficients

The symbols K, IK and KH used in Figure 9 are defined as follows:

K = reflection coefficients
IK = transmitted codes
KH = quantized reflection coefficients

The reflection coefficients K(1), K(2), K(3), K(4), K(5) and K(6) are coded using 6,5,4,4,4, and 4 bits, respectively. The coding and decoding tables are given in Table 2.

2.4.5 K-to-A conversion

The routine required for K-to-A conversion is the same as the one used in the 9.6 kb/s BBC coder [1], with the following input and output quantities:

KH = input quantized reflection coefficients (6 total)
AH = predictor coefficients for the quantized case (6 total)

2.4.6 Noise-Shaping Analysis

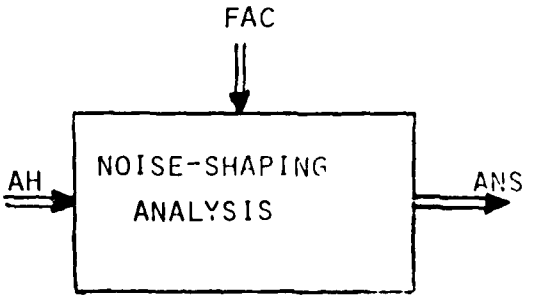


Figure 10. Noise shaping analysis

$$ANS(k) = (FAC)^{k} * AH(k), \quad 1 \leq k \leq 6$$

where

AH = predictor coefficients for the quantized case (6 total)
ANS = output coefficients (6 total)

The values of $(FAC)^k$ are:

$$FAC^1 = 0.684128$$
$$FAC^2 = 0.468032$$
$$FAC^3 = 0.320194$$
$$FAC^4 = 0.219054$$
$$FAC^5 = 0.149861$$
$$FAC^6 = 0.102524$$

2.5 Inverse Filter to Obtain the Second Residual

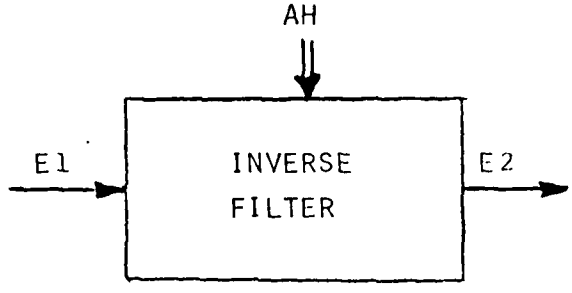


Figure 11. Inverse filter for the second residual

$$E2(n) = E1(n) + SUM[AH(i)*E1(n-i),\ 1 \le i \le 6]$$

where

E1 = input samples of the first residual (216 samples of present frame; 6 samples of past frame)
AH = predictor coefficients for the quantized case (6 total)
E2 = output samples of the second residual (216 total)

2.6 Gain Computation

The gain computation consists of the following steps: compute and code-decode energy (or mean-squared value) of the second residual, compute segment energies, compute the delta gains and code-decode them, and compute quantizer scale factors.

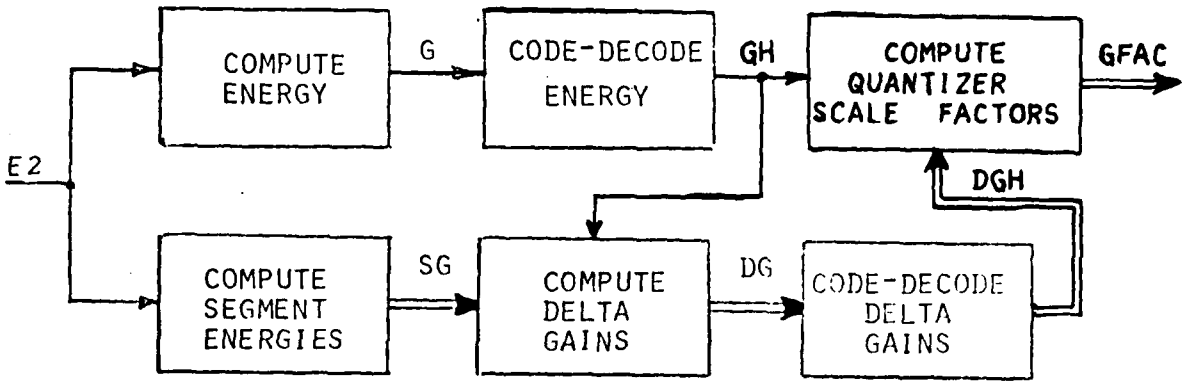


Figure 12. Block diagram of Gain Computation

The symbols used in Figure 12 are defined below:

E2 = second residual samples
G = energy of E2
GH = square root of the quantized energy
SG = segment energies
DG = delta gains
DGH = square root of the quantized delta gains
GFAC = quantizer scale factors

2.6.1 Compute Energy

$$G = 1/N * SUM[E2(i)^2, 1 \leq i \leq N], N=216$$

where

E2 = input samples of the second residual (216 total)
G = output energy

2.6.2 Code-Decode Energy

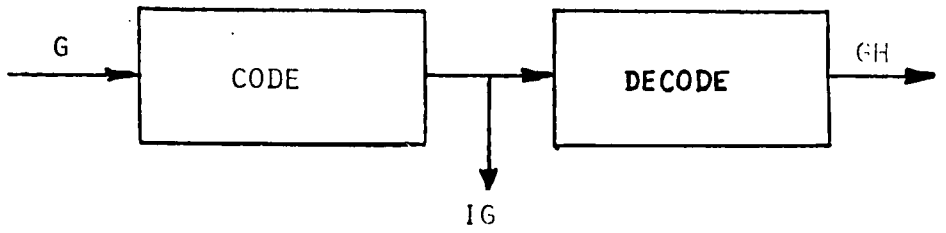


Figure 13. Encoding and decoding of energy

In Figure 13, we have

G = energy
IG = transmitted code
GH = square root of the quantized value

The energy G is coded using 6 bits. The coding and decoding tables are given in Table 3.

2.6.3 Compute Segment Energies

$$SG(j) = 1/72*SUM[E2(i)^2,\ (j-1)*72+1 \le i \le j*72], j=1,2,3$$

where

E2 = input samples of the second residual (216 total)
SG = segment energies (3 total)

2.6.4 Compute Delta Gains

$$DG(j) = SG(j)/(GH*GH), \quad j=1,2,3$$

where

DG = delta gains (3 total)
SG = segment energies (3 total)
GH = square root of the quantized energy

2.6.5 Code-Decode Delta Gains

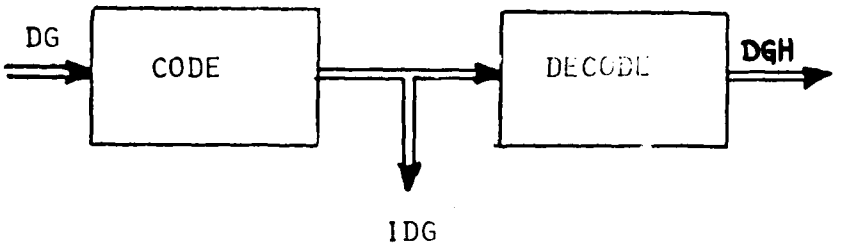


Figure 14. Encode and decode of the delta gains

In Figure 14, we have

DG = delta gains
IDG = transmitted codes
DGH = quantized values

The delta gains are coded using 2 bits each. Table 4 contains the coding and decoding tables for the delta gains.

2.6.6 Compute Quantizer Scale Factors

$$GFAC(j) = [GH*DGH(j)] \qquad j=1,2,3$$

where

DGH = quantized delta gains (3 total)
GH = square root of the quantized energy
GFAC = quantizer scale factors (3 total)

2.7 APC loop

For each input preemphasized speech sample SP(i), the following steps are performed.

Steps	1-4.	Compute predictions (4 total)
Step	5.	Compute APC residual, WO
Step	6.	Normalize APC residual
Steps	7,8.	Code-decode residual
Step	9.	Scale quantized residual
Steps	10-13.	Compute Q and update arrays; Q1,VH,RH

The output from each of these steps is marked in the block diagram of the APC loop, given in Figure 16, by a circled number; these numbers indicate the order in which the outputs are computed.

2.7.1 Compute Predictions for Sample i

(a) For noise shaping and spectral predictors (Steps 1,2,3)

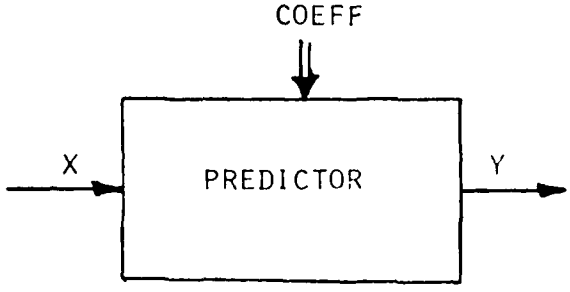


Figure 15. Noise shaping and spectral predictor in the APC loop

$Y = SUM[COEFF(j)*x(i-j), 1 \le j \le 6]$, for sample i

PREDICTOR TYPE	INPUT X	RANGE OF X	COEFF VECTOR	# OF COEFF	OUTPUT Y
Noise-shaping (zeros)	Q1(j)	$i-6 \le j \le i-1$	ZNS	6	QP
Noise-Shaping (poles)	Q1(j)	$i-6 \le j \le i-1$	AH	6	QP1
Spectral Predictor	VH(j)	$i-6 \le j \le i-1$	AH	6	VHP

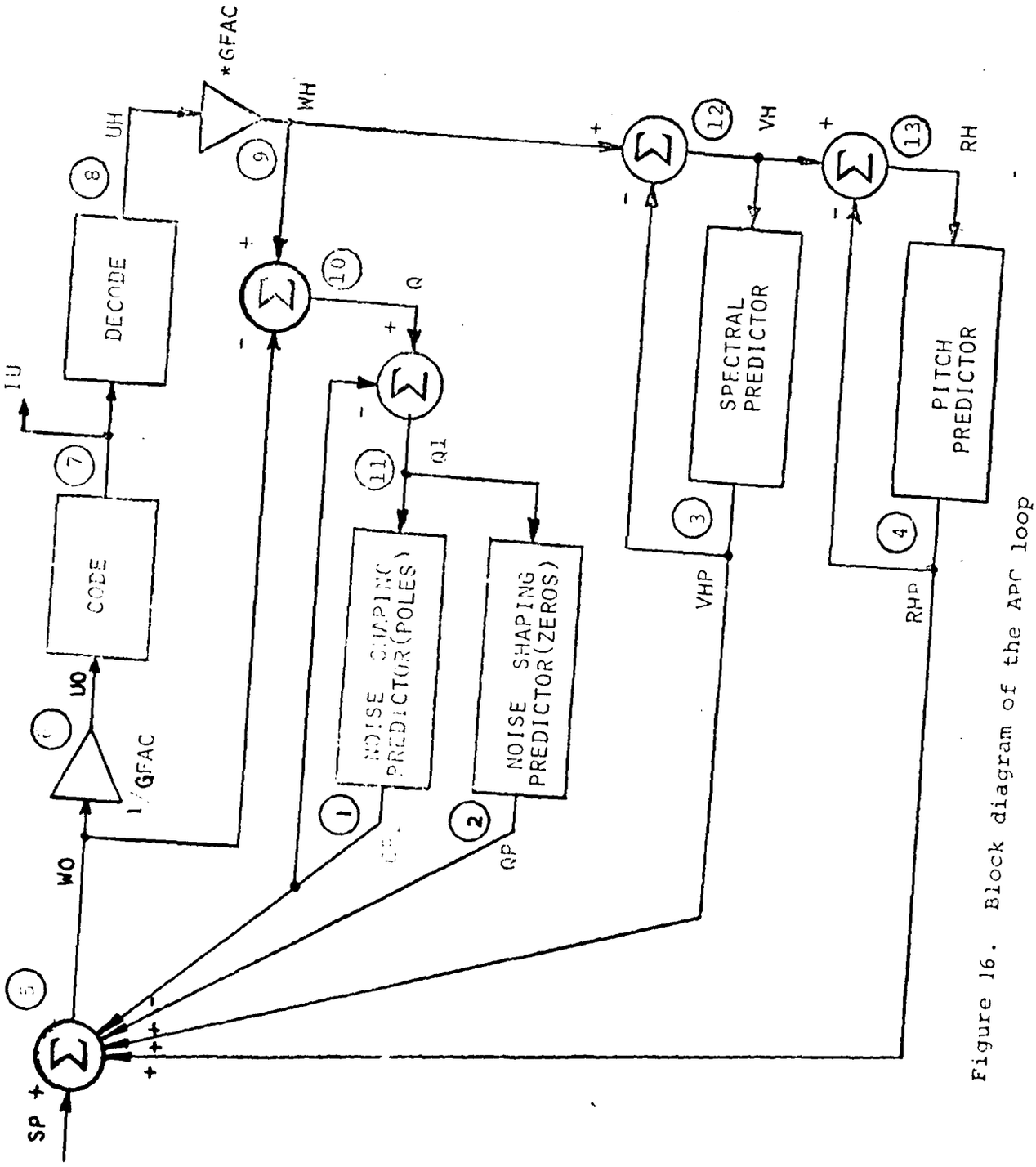


Figure 16. Block diagram of the APC loop

(b) For pitch predictor (Step 4)

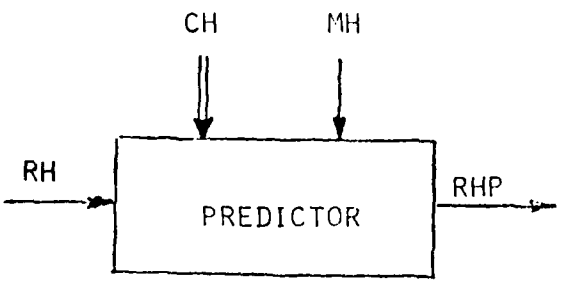


Figure 17. Pitch predictor in the APC loop

RHP = C1H*RH(i-MH+1) +C2H*RH(i-MH)+C3H*RH(i-MH-1), for sample i

where

RH = input samples (range from i-MH-1 to i-MH+1 for sample i
CH = quantized pitch predictor coefficients: C1H,C2H,C3H
MH = quantized pitch
RHP = output prediction

2.7.2 Compute APC Residual, WO (Step 5)

WO = SP(i) + QP - QP1 + RHP + VHP

2.7.3 Normalize APC Residual (Step 6)

UO = WO/GFAC(j)

where, for the i-th residual sample,

j=1 for $1 \leq i \leq 72$
j=2 for $73 \leq i \leq 144$
j=3 for $145 \leq i \leq 216$

2.7.4 Code-Decode Normalized Residual (Steps 7 and 8)

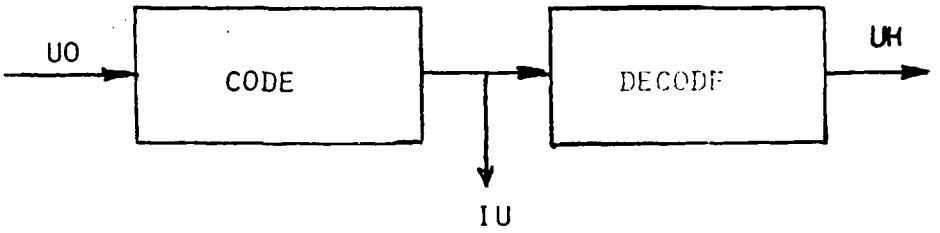


Figure 18. Encoding and decoding of the normalized residual in the APC loop

In Figure 18, we have

UO = normalized residual sample
IU = transmitted code
UH = quantized value

The normalized residual is coded using 2 bits. Table 5 gives the coding and decoding tables.

2.7.5 Scale the Quantized Residual (Step 9)

WH = UH*GFAC(j)

where, for the i-th sample,

j=1 for $1 \le i < 72$
j=2 for $73 \le i < 144$
j=3 for $145 \le i \le 216$

2.7.6 Update Arrays (Steps 10-13)

Arrays are updated in the following sequence:

(a) Q = WH - WO
(b) Q1(i) = Q - QP1
(c) VH(i) = WH - VHP
(d) RH(i) = VH(i) - RHP

Repeat the Steps 1 to 13 for each of the 216 input samples.

2.8 Folded Binary Code (FBC) for Encoding the Residual Samples

The FBC encoding may be performed as a separate operation or may be included as part of the coding table.

(a) Coding table approach: interchange the "J" values of 0 and 1 in Table 5.

(b) Separate operation: The residual is coded as in Table 5; then the codes are interchanged as in Figure 19. At the receiver the codes are again interchanged as in Figure 19 and then decoded as in Table 5.

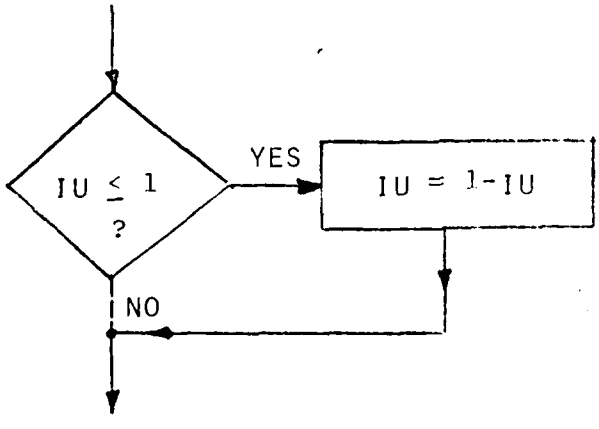


Figure 19. Flowchart for folded binary code (FBC); separate operation

3. RECEIVER

A block diagram of the receiver is given in Figure 20. In this section, we give specification for each of the receiver components.

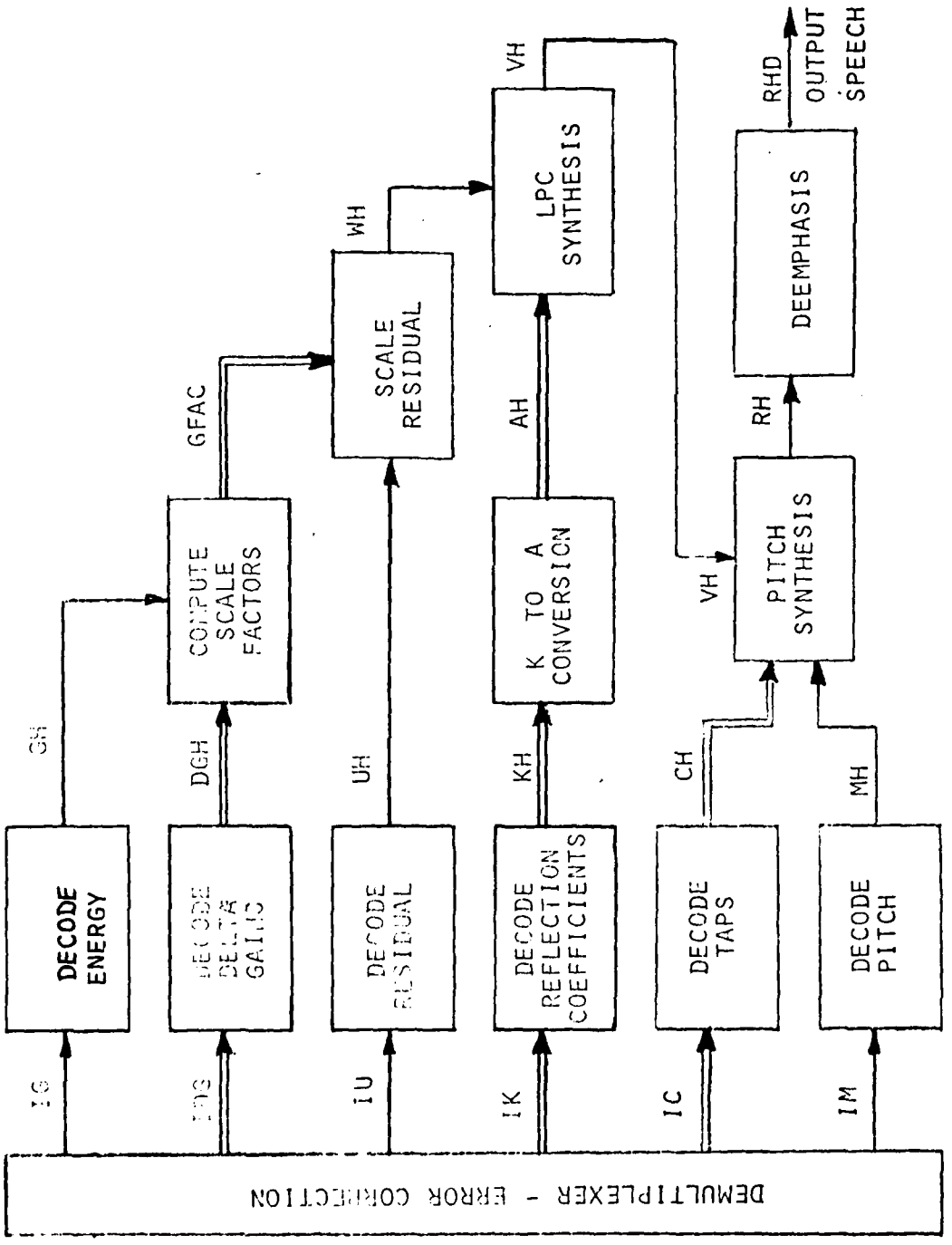


Figure 20. Block diagram of the receiver of the APC coder

3.1 Decode Parameters

Decode the received parameters using appropriate tables listed below.

Parameter	Tables
Energy	3
Delta Gains	4
Residual *	5
Reflection Coefficients	2
Pitch taps	1
Pitch	see Section 2.2.5

*Decode folded binary code (FBC) as shown in Fig. 19, if FBC encoding is done as a separate operation rather than as part of the coding table. (See Section 2.8).

3.2 Compute scale factors

See Section 2.6.6

3.3 Scale the quantized residual

See Section 2.7.5

3.4 K-to-A conversion

See Section 2.4.5

3.5 LPC synthesis

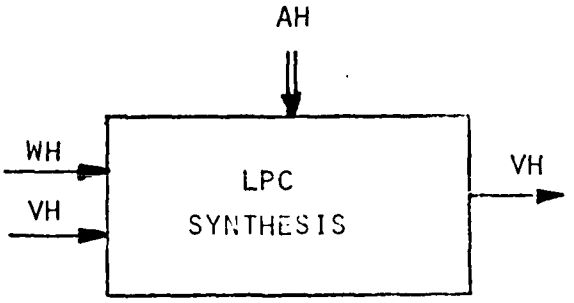


Figure 21. LPC synthesis

$$VH(i) = WH(i) - SUM[AH(j)*VH(i-j), 1 \leq j \leq 6], 1 \leq i \leq 216$$

where

WH = input residual samples (216 total)
AH = predictor coefficients (6 total)
VH = for each output sample i, six prior values (i-1 to i-6) of VH are required as input.

Save the last 6 samples of VH as initial condition of next frame. Also, the functions in Sections 3.4 and 3.5 can be combined into one, if lattice-form synthesis is used.

3.6 Pitch Synthesis

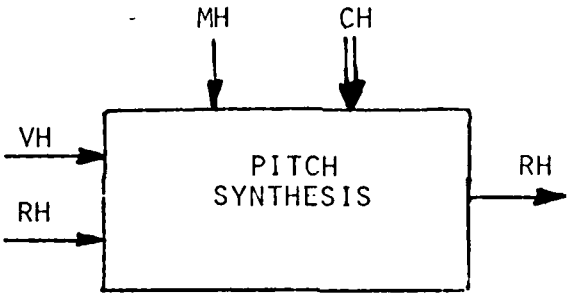


Figure 22. Pitch synthesis

$$RH(i) = VH(i) - C1H*RH(i-MH+1) - C2H*RH(i-MH) - C3H*RH(i-MH-1)$$

where

VH = input samples (216 total)
MH = pitch
CH = pitch predictor taps: C1H,C2H,C3H
RH = for each output sample i, three prior values: i-MH+1, i-MH,i-MH-1 of RH are required as input.

Save the last 134 samples of RH as initial condition of next frame.

3.7 Deemphasis

$$RHD(i) = RH(i) + ALPHA*RHD(i-1), \quad 1 \leq i \leq 216$$

where

RH = input samples (216 total)
ALPHA = constant = 0.4
RHD = output speech samples (216 total)

Save the last sample of RHD as initial condition of next frame. Note that RHD is the synthesized output speech.

REFERENCE

1. R. Viswanathan, J. Wolf, L. Cosell, K. Field, A. Higgins and W. Russell, "Design and Real-Time Implementation of a Baseband LPC Coder for Speech Transmission over 9600 BPS Noisy Channels", Final Report, BBN Report No. 4327, Vol. I and II, Bolt Beranek and Newman Inc., Cambridge, MA, February 1980.

Table 1(a)

CODING AND DECODING TABLES FOR PITCH TAPS

C1,C3 (3 Bits)

X(J)	J	R(J)
$-\infty$		
	0	-.488
-.427		
	1	-.366
-.305		
	2	-.244
-.183		
	3	-.122
-.061		
	4	.000
.061		
	5	.122
.183		
	6	.244
.305		
	7	.366
∞		

Table 1(b)

CODING AND DECODING TABLES FOR PITCH TAP
C2 (4 Bits)

X(J)	J	R(J)
-∞		
-.8851250Ø	Ø	-.91656251
-.81625Ø24	1	-.84968750
-.7[illegible]37499	2	-.78281250
-.[illegible]6249999	3	-.71593750
-.[illegible]1552499	4	-.64906249
-.[illegible]5437[illegible]93	5	-.58218749
-.[illegible]4615[illegible]7[illegible]93	6	-.51531249
-.[illegible]4[illegible]1[illegible]9[illegible]992	7	-.44843748
-.[illegible]3481[illegible]2496	8	-.38156248
-.[illegible]2812[illegible]499	9	-.3146374[illegible]
-.21[illegible]37499	10	-.24781249
-.14749999	11	-.18093749
-.[illegible]8[illegible]32499	12	-.11406249
-.[illegible]137[illegible]999	13	-.04718749
.[illegible]9312562	14	.01968752
∞	15	.08656252

Table 2(a)

CODING AND DECODING TABLES FOR THE REFLECTION COEFFICIENT K(1) (6 BITS)

X(J)	J	R(J)
-∞		
	0	-.98623391
-.98540065		
	1	-.98451736
-.98358107		
	2	-.98258864
-.98153679		
	3	-.98042203
-.97924066		
	4	-.97798879
-.97666232		
	5	-.97525692
-.97376801		
	6	-.97219076
-.97052009		
	7	-.96875066
-.96687680		
	8	-.96489257
-.96279173		
	9	-.96056771
-.95821358		
	10	-.95572206
-.95308557		
	11	-.95029607
-.94734517		
	12	-.94422411
-.94092366		
	13	-.93743417
-.93374563		
	14	-.92984749
-.92572883		
	15	-.92137819
-.91678373		
	16	-.91193306
-.90681341		
	17	-.90141150
-.89571356		
	18	-.88970540
-.88337240		
	19	-.87669948
-.86967121		

Table 2(a)
K(1) (cont.)

X(J)	J	R(J)
-.85448478	20	-.86227170
-.83768241	21	-.84629394
-.81912912	22	-.82863319
-.79868747	23	-.80915297
-.77621973	24	-.78771539
-.75159076	25	-.76418365
-.72467132	26	-.73842508
-.69534210	27	-.71031488
-.66349818	28	-.67974033
-.62905408	29	-.64660559
-.59194915	30	-.61083690
-.55215304	31	-.57238807
-.50967117	32	-.53124595
-.46454968	33	-.48743577
-.41687981	34	-.44102574
-.36680087	35	-.39213083
-.31450189	36	-.34091512
-.26022125	37	-.28759249
-.20424442	38	-.23242530
-.14689926	39	-.17572091
-.08854935	40	-.11782601
-.02958530	41	-.05911889
.02958537	42	.00000004
.08854942	43	.05911896

Table 2(a)
K(1) (cont.)

X(J)	J	R(J)
	44	.11782609
.14689934	45	.17572098
.20424450	46	.23242537
.26022132	47	.28759256
.31450195	48	.34091519
.36680095	49	.39213091
.41687988	50	.44102581
.46454975	51	.48743584
.50967123	52	.53124601
.55215311	53	.57238812
.59194921	54	.61083695
.62905414	55	.64660564
.66349822	56	.67974038
.69534215	57	.71031493
.72467136	58	.73842513
.75159079	59	.76418369
.77621976	60	.78771544
.79868749	61	.80915300
.81912915	62	.82863321
.83763244	63	.84629396
∞		

Table 2(b)

CODING AND DECODING TABLES FOR THE REFLECTION COEFFICIENT

K(2) (5 BITS)

X(J)	J	R(J)
$-\infty$		
	0	-.74930506
-.73101083		
	1	-.71160168
-.69104202		
	2	-.66929988
-.64634765		
	3	-.62216299
-.59672958		
	4	-.57003794
-.54208627		
	5	-.51288126
-.48243376		
	6	-.45078440
-.41795417		
	7	-.38399474
-.34896366		
	8	-.31292941
-.27597119		
	9	-.23817849
-.19965046		
	10	-.16049504
-.12032789		
	11	-.08077108
-.04045165		
	12	.00000000
.04045156		
	13	.08077099
.12032781		
	14	.16049496
.19965037		
	15	.23817841
.27597111		
	16	.31292932
.34896358		
	17	.38399466
.41795409		
	18	.45078433
.48243369		
	19	.51288119
.54208621		
	20	.57003786
.59672952		

Table 2(b)
K(2)(cont.)

X(J)	J	R(J)
	21	.62216294
.64634760		
	22	.66929983
.69104198		
	23	.71160164
.73101079		
	24	.74930502
.76652286		
	25	.78270507
.79789401		
	26	.81213313
.82546644		
	27	.83793808
.84959197		
	28	.86047144
.87061901		
	29	.88007615
.88888308		
	30	.89707866
.90470025		
	31	.91178370
∞		

Table 2(c)

CODING AND DECODING TABLES FOR REFLECTION COEFFICIENT

K(3) (4 BITS)

X(J)	J	R(J)
- ∞		
	0	-.75245864
-.72465231		
	1	-.69433287
-.66129490		
	2	-.62547535
-.58680991		
	3	-.54526938
-.50086620		
	4	-.45366014
-.40376312		
	5	-.35134251
-.29662241		
	6	-.23988272
-.18145542		
	7	-.12171834
-.06108626		
	8	.00000000
.06108626		
	9	.12171833
.18145541		
	10	.23988271
.29662241		
	11	.35134250
.40376312		
	12	.45366013
.50086619		
	13	.54526937
.58680990		
	14	.62547535
.66129489		
	15	.69433286
∞		

Table 2(d)

CODING AND DECODING TABLES FOR REFLECTION COEFFICIENT

K(4) (4 BITS)

X(J)	J	R(J)
- ∞		
	0	-.57005347
-.53250506		
	1	-.49273689
-.45079815		
	2	-.40677000
-.36076742		
	3	-.31293977
-.26347055		
	4	-.21257565
-.16050062		
	5	-.10751649
-.05391451		
	6	.00000000
.05391451		
	7	.10751648
.16050061		
	8	.21257564
.26347053		
	9	.31293977
.36076741		
	10	.40677000
.45079814		
	11	.49273689
.53250505		
	12	.57005347
.60536267		
	13	.63843988
.66931576		
	14	.69804088
.72468230		
	15	.74932021
∞		

Table 2(e)

CODING AND DECODING TABLES FOR REFLECTION COEFFICIENT

K(5) (4 BITS)

X(J)	J	R(J)
- ∞		
	0	-.50890809
-.47343143		
	1	-.43634928
-.39772316		
	2	-.35763384
-.31618157		
	3	-.27348579
-.22968443		
	4	-.18493246
-.13940015		
	5	-.09327063
-.04673719		
	6	.00000000
.04673717		
	7	.09327062
.13940014		
	8	.18493246
.22968442		
	9	.27348579
.31618155		
	10	.35763383
.39772316		
	11	.43634928
.47343142		
	12	.50890808
.54273631		
	13	.57489084
.60536267		
	14	.63415766
.66129489		
	15	.68680496
∞		

Table 2(f)

CODING AND DECODING TABLES FOR REFLECTIONCOEFFICIENT

K(6) (4 BITS)

X(J)	J	R(J)
- ∞		
	0	-.36516072
-.32298331		
	1	-.27948269
-.23480363		
	2	-.18910990
-.14258225		
	3	-.09541580
-.04781699		
	4	.00000000
.04781697		
	5	.09541579
.14258224		
	6	.18910989
.23480362		
	7	.27948267
.32298330		
	8	.36516071
.40589048		
	9	.44506935
.48261537		
	10	.51846758
.55258511		
	11	.58494609
.61554603		
	12	.64439614
.67152148		
	13	.69695902
.72075575		
	14	.74296685
.76365393		
	15	.78288344
∞		

Table 3

CODING AND DECODING TABLES FOR ENERGY (6 BITS)

X(J)	J	R(J)
0.12232071	0	0.33256382
0.14962357	1	0.36781124
0.18302061	2	0.40679443
0.22387212	3	0.44990933
0.27384197	4	0.49759385
0.33496544	5	0.55033230
0.40973211	6	0.60866034
0.50118723	7	0.67317039
0.61305579	8	0.74451765
0.74989421	9	0.82342681
0.91727594	10	0.91069929
1.12201344	11	1.00722152
1.37246095	12	1.11397386
1.67887398	13	1.23204054
2.05352500	14	1.36262074
2.51168536	15	1.50704074
3.07255733	16	1.66676740
3.75637393	17	1.84342298
4.59726977	18	2.03880173
5.62341309	19	2.25488809
	20	2.49387682

Table 3
ENERGY (cont.)

X(J)	J	R(J)
6.o7859981		
	21	2.75819513
8.41395115		
	22	3.05052787
10.27200480		
	23	3.37384397
12.58925340		
	24	3.73142737
15.39926480		
	25	4.12690991
16.83649020		
	26	4.56430846
23.04092360		
	27	5.04806560
25.15382810		
	28	5.58309466
34.47465850		
	29	6.17482972
42.16954320		
	30	6.82928115
51.56221390		
	31	7.55309576
63.09572790		
	32	8.35362506
77.17914580		
	33	9.23900044
94.46660220		
	34	10.21821390
115.47818800		
	35	11.30121100
141.25374100		
	35	12.49899180
172.76258700		
	37	13.82372160
211.34688530		
	38	15.28885570
258.52346400		
	39	16.90927510
316.22775300		
	40	13.70143700
386.51241900		
	41	20.58354540
473.15122200		
	42	22.87573150

Table 3
ENERGY (cont.)

X(J)	J	R(J)
578.76192500	43	25.30025960
737.94572500	44	27.93175740
865.95427900	45	30.94745760
1059.25352000	46	34.22748420
1295.63658000	47	37.35515120
1584.89297000	48	41.36730190
1938.65242000	49	46.33468990
2371.37350000	50	51.21238140
2900.68085000	51	56.54022640
3548.13379000	52	62.54335060
4340.10205000	53	69.28272910
5308.84361000	54	76.52579440
6493.21567000	55	84.74712280
7943.28168000	56	93.72921370
9716.27893000	57	103.66328700
11885.32250000	58	114.65023700
14537.84370000	59	126.90167400
17782.79220000	60	140.24098800
21752.03560000	61	155.10471300
26607.24850000	62	171.54377700
32545.17310000	63	189.72517200

Table 4

CODING AND DECODING TABLES FOR DELTA GAINS (2 BITS EACH)

X(J)	J	R(J)
-∞		
	0	0.48977882
0.43651563		
	1	0.79432824
0.09125794		
	2	1.12201844
1.05958589		
	3	1.49623564
∞		

Table 5

CODING AND DECODING TABLES FOR RESIDUAL SAMPLES (2 BITS)

(NATURAL BINARY CODE)

X(J)	J	R(J)
-∞		
	0	-1.50489251
-0.97617319		
	1	-0.45145388
[illegible]		
	2	0.45145388
0.97617319		
	3	1.50489251
∞		

APPENDIX B

USER'S GUIDE
FOR THE FORTRAN SIMULATION OF
THE BBN'S 16 kb/s APC CODER

August 1980

Prepared by

W. Russell, A. Higgins, and R. Viswanathan
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, Massachusetts 02138

TABLE OF CONTENTS

1. INTRODUCTION

This guide provides information necessary to use the FORTRAN simulation of the BBN 16 kb/s APC coder. Installation of this simulation on the user's computer system will require some software modifications. These modifications are specified in detail in Section 2 of this guide. In Section 3, a typical user session is described. Section 4 outlines how the user may alter the operation of the coder by resetting various flags and coder parameters. The simulation of the coder operating in the presence of channel bit-errors is discussed in Section 5.

2. SIMULATION SOFTWARE

The simulation software consists of a main program 16KMN, and the following five subroutine packages:

1. 16KIO - File handling and data I/O routines
2. 16KGEN - General utility routines
3. 16KCOD - Quantization, encoding and decoding routines
4. 16KER - Channel bit-error simulation, error-protection, and error-correction routines
5. 16KPR - All other routines

The FORTRAN listings of the main program and of each of the five subroutine packages are given in Appendix C. The simulation also uses one routine from the IBM scientific subroutine package (Routine NDTR for evaluating the normal distribution function, called by the subroutine OPTQ in the 16KCOD package) and several routines from our BBN speech library package. The FORTRAN code for these latter routines is not included in the supplied software, since they have been designed specifically for the BBN computer system. For the user's reference, a list of these routines from the BBN speech library, their calling sequence and a brief description of their purpose are given at the end of this section.

The user must substitute his own software to perform the tasks of the missing routines. The locations within the main module

16KMN, where substitutions should be made, are specially marked with a string of asterisks and comments. The steps required to perform the substitutions are listed below.

1. Speech I/O

The BBN simulation system employs disk files for speech I/O, in that digitized speech samples are read into a buffer from an input disk file and processed speech samples are written out from a buffer into an output disk file. (The input and output disk files may be compared using a separate D/A playout program, which is not part of the simulation software.) The following parts of the main program 16KMN have to be modified to suit the user's I/O facility.

a) Specification statements for file handling: The specification statements at the top of the main program labeled "DATA FOR FILE HANDLING" should be replaced with appropriate ones that may be needed for the user's specific speech I/O.

b) Opening input and output speech files: The user must replace the code labeled "OPEN INPUT AND OUTPUT SPEECH FILE" below statement 100 in the main program and the subroutines OPNIF and OPNOF in the 16KIO package, with his own software to provide access for input and output speech samples. Also, at this place in the main program, the quantity NFRAME (number of samples/frame) must be computed from the sampling frequency in Hz, FREQS and the frame size in ms, TFRAME. (The BBN-specific subroutine OPNIF reads in the value of FREQS from the header of the input speech file, or allows the user to specify it in the case of an unheadered file.)

c) Reading in speech samples: The code labeled "READ IN NFRAME SAMPLES" after statement 2000 in the main program and the subroutine ISAMP in the 16KIO package should be replaced with the user's own code to read in NFRAME number

of speech samples. These samples should be stored as floating-point numbers in the buffer SPEECH starting from the location N2S. The user's code must also check for the end of the input speech data. When the end is detected, the program control should be transferred to statement 2010.

d) Writing-out speech samples: The user must replace the code labeled "OUTPUT SAMPLES" below statement 4000 in the main program and the subroutine OSAMP in the 16KIO package, with his own code to write out NFRAME number of output speech samples from the buffer SLAST, starting from location N2S.

e) Closing input and output speech files: The user must replace the code labeled "CLOSE FILES" around statement 4050 with his own code to close the input and output access.

2. FFT of Real Data

a) The subroutine PITCH in the package 16KPR calls another subroutine FFTR to perform FFT of real data. A description of FFTR is given at the end of this section. The user must replace FFTR with his own subroutine.

b) A related subroutine WRWI is called by the main program (after statement 5 at the top of the program) to set up cosine table to be used by FFTR. The user must either remove this call or replace with another depending upon how his own FFTR subroutine is organized.

3. Random Number Generation

a) The subroutine ERRCHN in the package 16KER calls another subroutine RANDOM to generate pseudo-random numbers. A description of RANDOM may be found at the end of this section. Again, the user must replace RANDOM with his own subroutine.

b) A related subroutine ZETRAN is called by the main program (after statement 10 at the top of the program) to initialize the random-number generator at a prespecified point. This is necessary if one wants to employ an identical sequence of random numbers in two separate experiments. Again, the user must either remove this call or substitute ZETRAN with his own subroutine.

Software Change to Suit Different Input-Speech Wordlengths

The BBN 16 kb/s APC coder has been designed with the assumption that its input is 11-bit (including the sign bit) linear PCM speech. (To avoid a possible confusion, note that we store input speech samples using 12 bits each after extending or duplicating the sign bit to the left, and that three such 12-bit samples are packed in one 36-bit computer word.) If the user plans to use a different linear PCM speech as coder input, he must change the gain quantization ranges in dB, GMAX and GMIN, in the subroutine QTBLG (16KCOD routine package), to ensure proper gain quantization. This is accomplished by setting the value of the quantity DBCHANG, specified via a DATA statement in the same subroutine, to be equal to 6 times (actual speech sample size in bits -11). The factor 6 is due to the 6 dB/bit rule. For example, if the coder input is 9-bit linear PCM speech, then DBCHANG = -12.0.

A LIST OF SUBROUTINES FROM THE BBN SPEECH LIBRARY

```
INIT - CLOSES ALL OPEN FILES AND INITIALIZES SYSTEM
        CALL INIT

OPENIF - OPENS FILE AS INPUT FILE
         CALL OPENIF(JFN,BYTSIZ)
         CALL OPENIF(JFN,BYTSIZ,FILNAM)
         CALL OPENIF(JFN,BYTSIZ,FILNAM,IERR)
JFN   = JOB FILE NUMBER, RETURNED BY SUBROUTINE
BYTSIZ = FILE BYTE SIZE
FILNAM = POINTER TO FILE NAME.  IF THIS ARGUMENT IS 0 OR NOT
        GIVEN THEN THE FILE NAME IS TO BE TYPED IN.
IERR = OPTIONAL ERROR STATE ARG.  IF NOT GIVEN, FILE OPENING
        ERRORS WILL BE HANDLED BY THE IO ERROR HANDLER.  IF THIS
        ARG IS GIVEN, THEN THIS SUBROUTINE WILL ALWAYS RETURN,
        WITH IERR=0 IF THE FILE WAS OPENED.  IF THE FILE WASN'T
        OPENED, THEN IERR=JSYS ERROR CODE AND RETURNED JFN = -1.

OPENOF - OPEN FILE AS OUTPUT FILE
          CALL OPENOF(JFN,BYTSIZ)
          CALL OPENOF(JFN,BYTSIZ,FILNAM)
          CALL OPENOF(JFN,BYTSIZ,FILNAM,IERR)
ARGUMENTS SAME AS FOR OPENIF

CLOSF - CLOSES FILE, GIVEN JFN
        CALL CLOSF(JFN,NOREL)
NOREL = OPTIONAL ARGUMENT: IF GIVEN AND NONZERO, THE FILE IS
        CLOSED WITHOUT RELEASING THE JFN. IF ZERO OR NOT GIVEN,
        THE FILE IS CLOSED AND/OR THE JFN RELEASED, AS
        APPROPRIATE
FILNAM -   GETS FILE NAME, GIVEN JFN
           CALL FILNAME (JFN,ARRAY)
JFN     = JOB FILE NUMBER OF FILE
ARRAY   = POINTER TO ARRAY WHERE FILE NAME IS TO BE STORED

SFBSZ  - (RE) SETS FILE BYTE SIZE
          CALL SFBSZ (JFN,IBSIZE)
JFN     = JOB FILE NUMBER
IBSIZE = NEW BYTE SIZE

SFPTR  -  SETS FILE POINTER
          CALL SFPTR (JFN,NBYTE)
JFN      = JOB FILE NUMBER
NBYTE    = BYTE NO. TO WHICH POINTER IS TO BE SET
         =-1, WILL POINT TO CURRENT END OF FILE
```

```
RFPTR - READS FILE POINTER
        CALL RFPTR(JFN,NBYTE)
JFN     = JOB FILE NUMBER
NBYTE   = BYTE NUMBER OF POINTER IN FILE


SINB - STRING INPUT FROM FILE WITH ARBITRARY BYTE SIZE
       CALL SINB(EOF,JFN,POINTR,BYTSIZ,NBYTE)
       CALL SINB(EOF,JFN,POINTR,BYTSIZ,NBYTE,ENDCHR)
EOF    =ASSIGNED STATEMENT NO. FOR END OF FILE TRANSFER
JFN    =JOB FILE NUMBER
POINTR = POINTER TO ARRAY WHERE STRING IS TO BE STORED
BYTSIZ   = BYTE SIZE IN ADDRESR SPACE; IT CAN BE DIFFERENT FROM
             FILE BYTE SIZE.  BYTE IS ALWAYS RIGHT-JUSTIFIED WITH
             EXTRA ZEROS TO THE LEFT OR TRUNCATION IF NECESSARY
             DEPENDING ON THE RELATION BETWEEN THE TWO BYTE SIZES.
NBYTE   = NUMBER OF BYTES ACCORDING TO THE FOLLOWING:
          =0, ZERO BYTE TERMINATES
          >0, EXACT BYTE COUNT
          <0, NEGATIVE BYTE COUNT OR A BYTE OF -1, WHICHEVER COMES
          FIRST.
ENDCHR  = OPTIONAL RIGHT JUSTIFIED BYTE ON WHICH TO TERMINATE INPUT.
          OVERRIDES -1 TERMINATION WHEN NBYTE<0

SOUTB - STRING OUTPUT TO FILE, ARBITRARY BYTE SIZE
        CALL SOUTB(JFN,POINTR,BYTSIZ,NBYTE)
POINTR = POINTER TO ARRAY FROM WHICH STRING IS OUTPUT
OTHER ARGUMENTS SAME AS IN SINB.

PSOUT - ASCII STRING(S) OUTPUT TO TTY
        CALL PSOUT(POINTR1,POINTR2,...)
CALLS ASCZA IF HOLLERITH ARGUMENT

ASCZA - SEARCHES A 7-BIT STRING FOR A ZERO WORD.  THEN
TRACES BACK LOOKING FOR A WORD WITH AN '&' AND THE REST
FILLED WITH BLANKS.  IT WILL ONLY SKIP BACK OVER WORDS WHICH ARE
ALL SPACES.
IF FOUND, THE '&' AND THE ALL
BLANKS ARE REPLACED WITH NULLS.  IS USEFUL FOR FORTRAN LITERALS.
IF (OPT) 2ND ARG IS GIVEN IT IS A LEFT JUSTIFIED TERMINATOR BYTE
TO BE USED INSTEAD OF '&'.  IF IT IS SPACE, THEN
THIS MEANS TO DELETE ALL TRAILING SPACES BEFORE THE ZERO WORD.
     CALL ASCZA(STRING,TERM)
 STRING = A 7-BIT STRING WHICH MUST BE TERMINATED BY A ZERO WORD
```

```
 USUALLY A HOLLERITH.

PSOUTR - ASCII STRING(S) OUTPUT TO TTY FOLLOWED BY CR-LF
         CALL PSOUTR(POINTR1,POINTR2,...)
PUTS OUT ALL STRINGS, THEN A CR-LF.  CALLS ASCZA IF HOLLERITH
ARGUMENT

RALPH - READ ALPHANUMERIC STRING FROM TTY.
 ALLOWS CRTL-A OR RUBOUT EDITING.
 ALSO ALLOWS CTRL-R VIEWING OF THE STRING
 ALSO ALLOWS CTRL-U START OVER.
STRING IS TERMINATED BY CARRIAGE RETURN OR THE 400TH CHARACTER,
NEITHER OF WHICH IS PUT INTO THE ARRAY.
     CALL RALPH(ASCI,NCHAR)
ASCI = ARRAY IN WHICH STRING IS STORED WITH A NULL TERMINATOR
NCHAR   = NUMBER OF CHARACTERS IN THE STRING

LSH - LOGICAL SHIFT
       JFOO=LSH(WORD,NPLACES)
WORD = WORD TO BE SHIFTED
NPLACES = NUMBER OF LEFT SHIFTS (NEGATIVE IF TO BE A RIGHT SHIFT)

EXTFLT - SIGN-EXTENDS, THEN FLOATS, ASSUMING SIZE <= 27 BITS
        X=EXTFLT(IX,IEXWD)
IX        =WORD TO BE SIGN-EXTENDED
IEXWD      =1 IN THE MOST SIGNIFICANT BIT OF THE BYTE
        =LSH(1,BYTESIZE-1)

NRBYTS - FUNCTION TO COUNT BYTES IN A TERMINATED SPRING
         ICNT=NRBYTS(FROM,IDX,BYTSIZ,TERM)
               FROM = STRING ADDRESS (I.E., AN ARRAY ELEMENT)
               IDX = OPTIONAL STRING INDEX
                       IF ABSENT OR <= 0, DEFAULT VALUE OF 1 IS
                       USED.
                BYTSIZ  = OPTIONAL BYTE SIZE. IF ABSENT OR <= 0,
                             DEFAULT VALUE OF 7 IS USED.
                TERM = OPTIONAL TERMINATOR BYTE.  IF ABSENT,
                        DEFAULT VALUE OF 0 IS USED.
THE TERMINATOR BYTE IS NOT COUNTED.

CHMOVE - SUBROUTINE TO MOVE A CHARACTER STRING (NCHARS LONG)
CALLING SEQUENCE:
          CALL CHMOVE(FROM,IDX1,TO,IDX2,NCHARS)
          ARGUMENTS AS IN NRBYTS

ICHAR - FUNCTION WHICH RETURNS THE IDX-TH CHARACTER OF
 THE STRING CONTAINED AT "FROM", LEFT JUSTIFIED AND PADDED WITH
SPACES
```

```
 (SO IT CAN BE COMPARED WITH A FORTRAN SINGLE-CHARACTER LITERAL).
     J=ICHAR(FROM,IDX)
WHERE FROM AND IDX ARE ARRAY PTR AND INDEX, AS IN NRBYTS.

WRWI - Subroutine to generate cosine table required for the
subroutine FFTR
     CALL WRWI
CALCULATES 513 COSINES EQUALLY SPACED BETWEEN AND INCLUDING
0 AND 90 DEGREES

FFTR -- FFT of a Real Function
     CALL FFTR(LOG2N,NSAMP,S,TR,TI)
         COMPUTES THE (LOWER HALF + 1) OF THE FFT OF A REAL
         FUNCTION
         ARGUMENTS:
         LOG2N = LOG2(N) WHERE N IS THE ORDER OF THE FFT
               = MAXIMUM OF 10
         NSAMP = NUMBER OF REAL SAMPLES TO BE TRANSFORMED
         S     = VECTOR OF LENGTH NSAMP, CONTAINS SAMPLES
         TR    = VECTOR OF LENCTH N/2+1, REAL PART OF TRANSFORM
         TI    = VECTOR OF LENGTH N/2+1, IMAG PART OF TRANSFORM
         VECTORS S AND TR OR TI MAY BE IDENTICAL

ZETRAN - SETS THE RANDOM NUMBER "INITIAL VALUE" AND IS USED
TO SET THE ORIGIN OF THE RANDOM NUMBER SEQUENCE.
CALL ZETRAN(X,Y)
X=HIGH ORDER PART OF SEED
Y=LOW ORDER PART OF SEED

RANDOM - RANDOM REAL NUMBER GENERATOR
         GENERATES A RANDOM REAL
         NUMBER UNIFORMLY DISTRIBUTED BETWEEM TWO LIMITS.
         X =  RANDOM(A,B)
A=LOWER LIMIT
B=UPPER LIMIT
```

3. TYPICAL USER SESSION

The operation of the FORTRAN simulation requires only two inputs from the user: (1) a source of digitized input speech samples and (2) a location for the storage of the processed speech samples. At BBN, speech waveform samples are stored on disk files, as mentioned above. A typical user session, using disk file I/O, is described below. User input is underlined. In this session the input data file is <DCA16>BV1M.WAV and the output storage file is <DCA16>BV1M.TES. After inserting these two file names, the full coder simulation (transmitter and receiver) is executed without further intervention from the user. When all data has been processed, the program will print out the total number of frames processed and signal-to-quantization-noise (S/Q) ratios. The control of the program is then returned to the user. At that time the user may choose to process another speech utterance or abort the session.

Typical User Session

```
RUN 16KMN ÷

INPUT SPEECH FILE:  <DCA16>BV1M.WAV ÷
 10440 12-BIT SAMPLES AT 150 USEC = 1.5660 SECONDS
OUTPUT SPEECH FILE:  <DCA16>BV1M.TES.1 ÷

       FRAME COUNT = 48
       S/Q RATIO in dB:  LONG-TERM = 12.363; SEGMENTAL = 13.456
CONTINUE? (YES=-1,NO=0)=0 ÷

CPU TIME:  23.91 ELAPSED TIME: 1:35.95
NO EXECUTION ERRORS DETECTED
```

(Note that the symbol ÷ at the end of each input from the user denotes carriage return.)

4. SIMULATION OF SOME VARIATIONS OF THE OPTIMIZED APC CODER

The simulation has been designed to give the user the flexibility to modify the operation of the APC coder without software changes. Two methods of modification have been provided: (1) The user may set (True=-1) or clear (False=0) various flags that control the sequence of operations in the main program; and (2) The user may change the values of variables that specify important coder parameters. All flags and variables that the user may change are given their default values via DATA statements at the top of the main program 16KMN.

4.1 Flags

Flags have been provided so that the user may choose to keep or abort the execution of a specific section of the coder by setting or clearing the appropriate flag. For example, if the user wishes not to quantize the residual samples, he accomplishes this by simply clearing the flag IQ(1) (i.e., IQ(1)=0), prior to the execution of the coder. A list of the names of the flags and a description of the section of the coder each controls are given in Table 1. All flags, with the exception of ICHAN, ICHANE, and ICHANP, have their default value specified as True. The flags ICHAN, ICHANE and ICHANP are specified as False i.e., the coder is defaulted to operate in the absence of channel error.

Table 1. Flags in the FORTRAN simulation of the BBN 16 kb/s coder

Flag	Description
IPREM	Preemphasis - Deemphasis
KPPF	Pitch Prediction
INSTFX	3-Tap Pitch Predictor Stability Check
IHFCR	High Frequency Correction
KNSF	Noise Shaping
ISEGFG	Segmented Quantization
IFBC	Folded Binary Coding
ICHAN	Channel Simulation (Bit streaming)
ICHANE	Channel Error Simulation
ICHANP	Error Protection
IQ(1)-IQ(7)	Parameter Quantization
IQ(1	Residual Samples
IQ(2)	Energy
IQ(3)	Delta Gains
IQ(4)	Spectral Coefficients
IQ(6)	Pitch
IQ(7)	Pitch Predictor Taps
NOPRNT	Listing of quantization tables

4.2 Variables

A list of the variables defining coder parameters that may be modified by the user, and their default values are given in Table 2. This table also specifies the limits of parameter values within which the user may reset them without any software changes. Changes in the parameter values should be made in a manner that preserves the consistency of interdependent parameters such as TFRAME and NENSEC. Note also that choosing NPOLE > 6 requires specification of additional data, as indicated in Table 2 under NPOLE.

Table 2. A list of APC coder simulation variables and their description

	Variable Name	Description	Default Value
1.	FREQS	Sampling frequency (Hz) Sampling frequency must correspond to that of the input data	6666.66
2.	TFRAME	Interframe interval (ms) Set TFRAME such that NFRAME $\leq$ 300 where NFRAME = IFIXR (TFRAME * FREQS/1000.) NOTE: Check NENSEC when modifying TFRAME as it also depends on NFRAME Function IFIXR is given in the subroutine package 16KGEN	32.4
3.	BWF	Preemphasis bandwidth (Hz)	972.21465
4.	T40	Pitch extraction frame size (ms) Set T40 such that I40 $\leq$ 600 where I40 = IFIXR (T40*FREQS/1000.)	34.75
5.	F0L	Lower limit of pitch frequency (Hz) This parameter is used to compute the upper limit on pitch period IF0L, defined in samples, where IF0L = IFIXR ((FREQS/F0L)+.5)	50
6.	F0H	Upper limit of pitch frequency (Hz) This parameter is used to compute the lower limit on pitch period, IF0H, defined in samples where IF0H = IFIX(FREQS/F0H)	450
7.	LOG2P	FFT order (exponent of 2) 2**LOG2P $\geq$ I40+IF0L	9

Table 2
(cont.)

	Variable	Description	Default
8.	NTAPS	Number of pitch prediction taps 1 and 3 are permissible values, if no pitch prediction is required clear flag KPPF. Do not set NTAPS to zero.	3
9.	NPOLE	Number of poles for LPC analysis Buffers accomodate up to 14 coefficients. However, default values for quantization and channel error are specified for up to NPOLE=6 only. To operate the simulation with NPOLE >6, the user must make the following modifications: 1. Provide additional values for the coding ranges CMIN and CMAX in subroutine QTBLC (16KCOD Package) 2. Provide additional default values for the number of bits protected and the number of bits transmitted (arrays NPERC and NBITC respectively at the top of the main routine, 16KMN)	6
10.	NPHFC	Order of the computation of the minimum mean-squared prediction error used in the high frequency correction module HFCOR (package 16KPR) NPHFC $\leq$ NPOLE	2
11.	HFLAM	Scalar constant used in the high frequency correction module	0.035
12.	CHF	Autocorrelation coefficients of a high pass filter used in the high-frequency correction module 3 values required	+.375, -.25, +.0625
13.	BWANS	Noise shaping bandwidth (Hz)	800
14.	NENSEC	Number of segments used in the segmented quantization scheme. Set NENSEC such that NSMSEC is an integer where NSMSEC = NFRAME/NENSEC	3
15.	IRUNG	Switch to simulate rungs or stages of the phased real-time implementation. (IRUNG=Ø for full simulation; 1, for stage 1; 2, for stage 2; and 3, for stage 3).	Ø

Table 2
(cont.)

	Variable	Description	Default
16.	NTYP	Switch to set the type of distribution used in the optimal quantization of the residual samples 1 = Gamma, 2 = Exponential, 3 = Gaussian	3

The following parameters set the number of bits for quantization of the transmitted parameters. If quantization is not required for a specific parameter, clear the appropriate IQ flag -- do not set the parameter below to zero.

17.	NBITR	Number of bits for quantization of the residual samples Range - 1 to 3	2
18.	NBITG	Number of bits for quantization of the Energy Range - 1 to 7	6
19.	NBITSC	Number of bits for quantization of the delta gains Range - 1 to 3	2
20.	NBITC(I)	Number of bits for quantization of the spectral coefficient I Range - 1 to 7	6 5 4 4 4 4
21.	NBITP	Number of bits for quantization of the pitch Range - 2**NBITP ≥ IF0L - IF0H	7
22.	NBITT(I)	Number of bits for quantization of the pitch predictor taps Range = 1-5	3 4 3

5. CODER OPERATION WITH CHANNEL-ERROR SIMULATION

The FORTRAN simulation is defaulted to operate in the absence of channel error. To initiate the simulation of channel error on all transmitted parameters the user must set (true=-1) the two flags ICHAN and ICHANE. Also, the flag ICHANP must be set to invoke error protection of parameters. The coder is defaulted to operate at 1% channel error when these flags are set. The user may change the percentage of channel error and the amount of protection for each parameter independently by resetting their default values at the top of the main program. Table 3 gives the names of the variables that specify the number of high-order bits protected for each transmitted parameter. When changing the values of these variables, the user must keep in mind that the total number of protected bits for all transmitted parameters should be an integer multiple of 4. The percentage of channel error for each transmitted parameter is defined in the array ERP. In Table 4, the correspondence between the array entries and the transmitted parameters is defined.

Table 3. A list of transmitted parameters, along with the number of high-order bits of each that are error-protected

Variable	Transmitted Parameter	Default Value
NPERC (I)	Spectral Coefficients NPERC(I) $\leq$ NBITC(I)	5,4,3,2,2,2
NPERP	Pitch NPERP $\leq$ NBITP	7
NPERG	Energy NPERP $\leq$ NBITG	6
NPERSC	Delta gains NPERSC $\leq$ NBITSC	2
NPERT (I)	Pitch Predictor Taps NPERT(I) $\leq$ NBITT(I)	2,3,2

(No protection is provided for the residual samples.)

Table 4. Correspondence between the elements of the array ERP and the type of transmitted parameters. The value of the array element indicates the percentage of channel error due to which the corresponding transmitted parameter(s) are exposed.

Variable	Transmitted Parameters	Default Value
ERP (1)	Spectral Coefficients	0.01
ERP (2)	Pitch	0.01
ERP (3)	Energy	0.01
ERP (4)	Delta Gains	0.01
ERP (5)	Pitch Predictor Taps	0.01
ERP (6)	Residual Samples	0.01

APPENDIX C

A LISTING OF THE SOURCE PROGRAMS OF THE FORTRAN SIMULATION OF THE 16 KB/S APC CODER

```
C  16K BIT APC  CODER  :  MAIN PROGRAM
C
C*  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C*
C*              "DATA FOR FILE HANDLING"
C*
C  UTTERANCE DATA
      DIMENSION LSCR(600)
C*
      COMMON/WAVFIL/IWBSIZ,IEXND,IWBYT1
      COMMON/INREPT/INREPT
      COMMON/MAXAMP/MAXAMP,NOVER
      COMMON/NAME/TMPNAM(10),INAME(10),ONAME(10)
      COMMON/INDR/INDR(30),VBSNDR
      COMMON/NLPTS/NFLP1S
      DATA (INDR(I),I=1,30)/30,10,1,0,0,1,1,23**4000000000000/
      DATA JFRI/-1/,JFRO/-1/
C*
C*
C*  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C  GENERAL CODER
C ------------------
      DIMENSION SPEECH(600),S(600),SWD(600)
      DIMENSION R(600),RD(600),RR(600),RRR(600),RRES(600)
      DIMENSION GSEC(600)
      DIMENSION UU(600),VN(600),RN(600),Q(600),Q1(600)
      DIMENSION IR(600),IRR(600)
      DIMENSION SP(600),SLAST(600),SPQ(600)
C
      DATA MAXR/600/,NZERO/301/
      DATA FREQS/6666.66/,TFRAME/32.4/
      DATA NOPRNT/-1/,IRUNG/0/
C
      COMMON/FRMSIZ/NFRAME,NWIND
      COMMON/FREQS/FREQS,TSAMP
      COMMON/PI/PI,TWOPI
      COMMON/ALOG2/ALOG2
      COMMON/EXPN/WR(0/512)
C  ANALYSIS
C ----------
C    PREEMPHASIS
      DATA IPREM/-1/,BMF/972.21465/
C
C    PITCH EXTRACTION
      DIMENSION FR(512),FI(512),PM(512)
C
      DATA T40/39.75/,LOG2P/9/
      DATA PBL/50.0/,PBH/450.0/
C
C    PITCH PREDICTION
      DIMENSION B(3),BR(3)
```



```
C
      DATA KPPF/-1/,NTAPS/3/,INSTFX/-1/
C
      COMMON/TSCR/TSCR(3)
C
C    LPC ANALYSIS
      DIMENSION RSCR(15),C(14),A(14),SCR2(14)
      DIMENSION CTX(14),DC(14),BA(14)
      DIMENSION CR(14),AR(14)
C
      DATA NPOLE/6/,NPMAX/14/,ALPHA/0.54/
C
      COMMON/NPOLE/NPOLE
      COMMON/ALPHA/ALPHA,ALPH1
C
C    HIGH-FREQUENCY CORRECTION
      DIMENSION CHF(3)
C
      DATA IHFCR/-1/,NPHFC/2/,HFLAM/0.835/
      DATA CHF(1)/0.375/,CHF(2)/-0.25/,CHF(3)/0.0625/
C
C    NOISE SHAPING
      DIMENSION ANS(14)
C
      DATA KNSF/-1/,BWANS/800.0/
C
C    ENERGY
      DATA ENZERO/1.0E-7/
C
C    SEGMENTED QUANTIZATION
      DIMENSION ESTX(3),ESRX(3)
C
      DATA NENSEC/3/,ISECFG/-1/
C
      COMMON/NSEC/NENSEC,NSMSEC
C  QUANTIZATION
C --------
C
      DIMENSION IQ(7)
C
C    PITCH
      DATA NBITP/7/,IQ(6)/-1/
C
C    TAPS
      DIMENSION NBITT(3),NLEVT(3),XTAP(33,3),YTAP(32,3),TSTEP(3)
      DIMENSION IPTAP(3),IPTAPR(3)
C
      DATA NBITT(1)/3/,NBITT(2)/4/,NBITT(3)/3/
      DATA IQ(7)/-1/
C
```

```
C  <DCA16>16KMM.F4.2   Fri 21-Nov-80 9:36AM          Page 1:2

C    COEFF
       DIMENSION NBITC(14),NLEVC(14),XCOF(129,14),YCOF(128,14)
       DIMENSION CSTEP(14)
       DIMENSION ICU(14),ICUR(14)
C
       DATA NBITC(1)/6/,NBITC(2)/5/,NBITC(3)/4/
       DATA NBITC(4)/4/,NBITC(5)/4/,NBITC(6)/4/
       DATA IU(4)/-1/
C
C    OVERALL GAIN
       DIMENSION XG(129),YG(128)
C
       DATA NBITG/6/,IU(2)/-1/
C
C    DELTA GAINS
       DIMENSION XSEC(9),YSEC(8)
       DIMENSION IENS(3),IENSN(3)
C
       DATA NBITSC/2/,IU(3)/-1/
C
C    RESIDUAL
       DATA NBITR/2/,NTYP/3/,IQ(1)/-1/,IFBC/-1/
C
       COMMON/RESCOD/NLEVR,XRES(9),YRES(8)
C
C  CHANNEL ERROR SIMULATION AND PROTECTION
C  ---------------------------------------
       INTEGER SRANX,SRANY
C
       DIMENSION ERP(6),ISEQ(500),NPER(500),NBER(500)
       DIMENSION IDAT(500),ER(500),IDATS(500),IBS(500)
       DIMENSION ERN(50),IDATE(500),IDATR(500)
       DIMENSION INW(50),INWC(50),INWE(50),INWR(50)
       DIMENSION IB(500),IBR(500),NBSPA(600)
       DIMENSION NPERC(14),NPERT(3)
C
       COMMON/IMASK/IMASK(32)
       COMMON/IHAMMD/IHAMMD(8)
       COMMON/CUM/ICUMBT,IERCUM,NSERR
C
       DATA NOPT/3/,MAXER/500/
       DATA NWWT/50/,NONONE/1/
C  RANDOM NUMBER GENERATOR SEEDS
       DATA SRANX/3208359606/,SRANY/21743579315/
C  FLAGS TO CONTROL ERROR PROTECTION AND CORRECTION
       DATA ICHANE/0/,ICHAN/0/,ICHANP/0/
C  PERCENTAGE OF CHANNEL ERROR FOR EACH
C  TRANSMITTED PARAMETER TYPE DEFINED IN
C  ARRAY ERP - SEE USER'S GUIDE FOR CORRESPONDENCE
C  BETWEEN ARRAY ENTRY AND PARAMETER TYPE
       DATA ERP(1)/0.01/,ERP(2)/0.01/,ERP(3)/0.01/
       DATA ERP(4)/0.01/,ERP(5)/0.01/,ERP(6)/0.01/

C  <DCA16>16KMM.F4.2   Fri 21-Nov-80 9:36AM          Page 1:3

C  HAMMING WORD MASKS
       DATA IHAMMD(1)/4/,IHAMMD(2)/16/,IHAMMD(3)/32/
       DATA IHAMMD(4)/64/,IHAMMD(5)/1/,IHAMMD(6)/2/
       DATA IHAMMD(7)/8/,IHAMMD(8)/128/
C
C  AMOUNT OF PROTECTION FOR EACH PARAMETER
C  DEFINED IN VARIABLES NPER* - SEE USER'S
C  GUIDE FOR CORRESPONDENCE BETWEEN VARIABLE
C  AND PARAMETERS
       DATA NPERC(1)/5/,NPERC(2)/4/,NPERC(3)/3/,NPERC(4)/2/
       DATA NPERC(5)/2/,NPERC(6)/2/
       DATA NPERG/6/,NPERP/7/,NPENSC/2/
       DATA NPERT(1)/2/,NPERT(2)/3/,NPERT(3)/2/
^L
```

```
C
C    THIS IS THE RETURN POINT FOR
C    PROCESSING ANOTHER SPEECH INPUT FILE
C
100     CONTINUE
C
C
C*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C*
C*              INSERT ROUTINE TO
C*              "OPEN INPUT AND OUTPUT SPEECH FILE"
C*              SAMPLING FREQUENCY (FREQS) IN HZ AND FILE BYTE SIZE
C*              ARE OBTAINED FROM FILE HEADER IF IT EXISTS,OR FROM THE
C*              USER IF THE FILE DOES NOT HAVE A HEADER.
C*
C*              INITIALIZE FILE HANDLING
        CALL INIT
C*              OPEN INPUT FILE
        CALL OPNIF(IERIF,JFNI)
C*              CHECK FOR ERROR IN INPUT FILE OPEN
        IF(IERIF) GO TO 100
C*
C*              COMPUTE # OF SAMPLES/FRAME (NFRAME)
        FRM=(TFRAME*FREQS/1000.)
        NFRAME=IFIXR(FRM)
C*              EVALUATE # OF OUTPUT POINTS FOR THIS FILE
        NFRMT=IFIX(FLOAT(NFLPTS)/FLOAT(NFRAME))
        JNUM(5)=NFRMT*NFRAME
C*
200     CONTINUE
C*              OPEN OUTPUT FILE
        CALL OPNOF(IEROF,JFNO)
C*              CHECK FOR ERROR IN OUTPUT FILE OPEN
        IF(IEROF) GO TO 200
C*
C*
C*  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C
^L
```



```
C
C
C  *  *  *  *  PROGRAM STARTS HERE *  *  *  *
C
5       CONTINUE
C
C  INITIALIZE CONSTANTS
        ALOG2=ALOG(2.0)
        PI=4.0*ATAN(1.0)
        T2OPI=2.0*PI
C*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C*              INSERT ROUTINE TO
C*              "SET UP COSINE TABLE FOR FFT"
        CALL #Rnl
C*
C*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C
C  CREATE MASKS USED IN ERROR PROTECTION AND CORRECTION
        IF(.NOT.ICHAN) GO TO 100
        DO 10 I=1,32
        IMASK(I)=2**(I-1)
10      CONTINUE
C
C*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C*              INSERT ROUTINE TO
C*              "INITIALIZE RANDOM NUMBER GENERATOR"
        CALL ZETHAN(SHANX,SHANY)
C
C*  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
^L
```

```
C
C   SET INITIAL DATA FOR CODEK OPERATION
C
        ALPH1=1.0-ALPHA
C    PREEMPHASIS
        PEN=-ONE*TWOPI/FREQS
        PENFAC=EXP(PEN)
C    SET FRAME LOCATION AND ANALYSIS WINDOW
     (ANALYSIS WINDOW NOT USED IN THIS SIMULATION)
        NW1WD=NFRAME
        N2S=NZERO
        N2A=N2S+NFRAME-1
C    PITCH EXTRACTION
        F40=T40*FREQS/1000.
        I40=IFIX(F40)
        IF(I40.LT.NFRAME) I40=NFRAME
        I40EX=I40-NFRAME
        N2SP=N2S-I40EX
        IF0L=IFIX((FREQS/F0L)+.5)
        IF0H=IFIX(FREQS/F0H)
C    SEGMENTED QUANTIZATION
        NSNSEC=NFRAME/NENSEC
C    NOISE SHAPING
        PZFAC=-PI*ONEHUNDREDTHSAMP
        PZFAC=EXP(PZFAC)
C    PITCH QUANTIZATION
        NLEVP=2**NBITP
C    GENERATE QUANTIZATION TABLES FOR
C      PITCH TAPS
        CALL QTBLT(NTAPS,NBITT,NLEVT,RTAP,VTAP,TSTEP)
C      REFLECTION COEFFS
        CALL QTBLC(NPOLE,NBITC,NLEVC,RCOF,VCOF,CSTEP)
C      OVERALL ENERGY
        CALL QTBLG(NBITG,NLEVG,RG,VG,GSTEP)
C      DELTA GAINS
        CALL QTBLSC(NBITSC,NLEVSC,RSEC,VSEC)
C      RESIDUAL SAMPLES
        NLEVR=2**NBITR
        IFULR=NLEVR/2-1
        CALL QNTZ(NTAP,NLEVR,RRES,VRES)
C
C  PRINT QUANTIZATION DATA
        IF(NOPRNT) GO TO 299
        TYPE 297
297     FORMAT(1H ,10X,34HTYPE QUANTIZATION TABLES - STARTED)
298     CONTINUE
        DO 202 I=1,NTAPS
        PRINT 201
201     FORMAT(1H1)
        PRINT 200,I
200     FORMAT(1H ,20X,6HTAP # ,I2,/)
        PRINT 202
```



```
202     FORMAT(1H ,12X,4HR(J),12X,1HJ,18X,4HR(J),/)
        PRINT 204,RTAP(1,I)
204     FORMAT(1H ,10X,F10.8)
        DO 206 J=1,NLEVT(I)
        JJ=J-1
        PRINT 208,JJ,VTAP(J,I)
208     FORMAT(1H ,27X,I3,7X,F10.8)
        PRINT 204,RTAP(J+1,I)
206     CONTINUE
202     CONTINUE
C  COEFF
        DO 220 I=1,NPOLE
        PRINT 201
        PRINT 200,I
204     FORMAT(1H ,27X,8HCOEFF # ,I2,/)
        PRINT 202
        PRINT 204,RCOF(1,I)
        DO 211 J=1,NLEVC(I)
        JJ=J-1
        PRINT 208,JJ,VCOF(J,I)
        PRINT 204,RCOF(J+1,I)
211     CONTINUE
220     CONTINUE
C  GAIN
        PRINT 201
        PRINT 206
206     FORMAT(1H ,30X,4HGAIN,/)
        PRINT 202
        PRINT 222,RG(1)
222     FORMAT(1H ,10X,F14.8)
        DO 224 J=1,NLEVG
        JJ=J-1
        PRINT 226,JJ,VG(J)
        PRINT 222,RG(J+1)
226     FORMAT(1H ,31X,I3,7X,F14.8)
224     CONTINUE
C  DELTA GAINS
        PRINT 201
        PRINT 208
208     FORMAT(1H ,26X,11HDELTA GAINS,/)
        PRINT 202
        PRINT 222,RSEC(1)
        DO 234 J=1,NLEVSC
        JJ=J-1
        PRINT 226,JJ,VSEC(J)
        PRINT 222,RSEC(J+1)
234     CONTINUE
C  RESIDUAL
        PRINT 201
        PRINT 209
```

```
289     FORMAT(1H ,2HR,8HRESIDUAL,/)
        PRINT 282
        PRINT 222,RRES(1)
        DO 244 J=1,NLEVR
        JJ=J-1
        PRINT 226,JJ,PRES(J)
        PRINT 222,RRES(J+1)
244     CONTINUE
        TYPE 296
296     FORMAT(1H ,10X,9HTYPE DONE)
299     CONTINUE
C
C
C
C   SET UP PARAMETERS REQUIRED FOR CHANNEL ERROR SIMULATION
C     TOTAL NUMBER OF BITS PER TRANSMITTED PARAMETER STORED IN NBER
C     TOTAL NUMBER OF PROTECTED BITS FOR EACH TRANSMITTED PARAMETER
C       STORED IN NPER
C     SEQUENCE OF TRANSMITTED PARAMETERS STORED IN ISEQ
C       (SEE NOTE  FOR CORRESPONDENCE BETWEEN PARAMETER
C       NUMBER AND PARAMETER TYPE)
C
        IF(.NOT.ICHAN) GO TO 350
C     ORDER OF SEQUENCE DEFINED BY K
        K=0
C     SPECTRAL COEFFS
        DO 315 I=1,NPOLE
        K=K+1
        NBER(K)=NBITC(I)
        ISEQ(K)=K
        NPER(K)=NPERC(I)
315     CONTINUE
C     OVERALL ENERGY
        K=K+1
        ISEQ(K)=16
        NBER(K)=NBITG
        NPER(K)=NPERG
C     PITCH
        K=K+1
        ISEQ(K)=15
        NBER(K)=NBITP
        NPER(K)=NPEPP
C     DELTA GAINS
        L=20
        DO 320 I=1,NFRSEC
        K=K+1
        L=L+1
        ISEQ(K)=L
        NBER(K)=NBITSC
        NPER(K)=NPERSC
320     CONTINUE
```



```
C     PITCH TAPS
        L=30
        IF(NTAPS.GT.1) GO TO 322
        L=L+1
        K=K+1
        ISEQ(K)=L
        NBER(K)=NBITT(1)
        NPER(K)=NPERT(1)
        GO TO 330
322     DO 325 I=1,NTAPS
        K=K+1
        L=L+1
        ISEQ(K)=L
        NBER(K)=NBITT(I)
        NPER(K)=NPERT(I)
325     CONTINUE
330     CONTINUE
        NDATP=K
C  IF NO ERROR PROTECTION SET NPER=0
        IF(ICHANP) GO TO 335
        DO 333 I=1,NDATP
        NPER(I)=0
333     CONTINUE
335     CONTINUE
C  SET START POSITION IN SEQUENCE FOR RESIDUAL SAMPLES
        NSTRES=K+1
C  COMPUTE TOTAL BITS FOR RESIDUAL SAMPLES
        NRBIT=NBITR*NFRAME
        NDATT=NDATP+NRBIT
C   SET ISEQ.... FOR RESIDUAL SAMPLES
        L=35
        DO 340 I=NDATP+1,NDATT
        L=L+1
        ISEQ(I)=L
        NPER(I)=0
        NBER(I)=NUMONE
340     CONTINUE
C    SET UP BIT ARRAY FOR RESIDUAL SAMPLES (BIT STREAMING)
        DO 343 I=N2S,N2E
        NBSPA(I)=NBITR
343     CONTINUE
C
C   INITIALIZE CUMULATIVE ERROR COUNTS
        ITCUMB=0
        ITCUME=0
        NTSERR=0
C
350     CONTINUE
C
```

```
C  ZERO ARRAYS AND S/Q RATIO DATA
        NAPCT=0
        DAPSU=0.0
        EAPSU=0.0
        DAPSQU=0.0
        EAPSQU=0.0
        CALL SETZM(S,MAXN)
        CALL SETZM(SPEECH,MAXR)
        CALL SETZM(SMO,MAXN)
        CALL SETZM(R,MAXR)
        CALL SETZM(RO,MAXR)
        CALL SETZM(RM,MAXR)
        CALL SETZM(VM,MAXR)
        CALL SETZM(Q,MAXR)
        CALL SETZM(U1,MAXR)
        CALL SETZM(UO,MAXR)
        CALL SETZM(PR,MAXR)
        CALL SETZM(RRR,MAXR)
        CALL SETZM(RRES,MAXR)
        CALL SETZM(SP,MAXR)
        CALL SETZM(SLAST,MAXR)
```



```
C
C  *  *  *  *  *  *  TRANSMITTER *  *  *  *  *  *
C  INITIALIZE FRAME COUNT
        NF=0
C  RETURN POINT FOR NEXT FRAME
2000    CONTINUE
C  UPDATE FRAME COUNT
        NF=NF+1
C  PUSH DOWN ARRAYS
        CALL PSHDN(SPEECH,MAXR,NFRAME)
        CALL PSHDN(S,MAXN,NFRAME)
        CALL PSHDN(SMO,MAXN,NFRAME)
        CALL PSHDN(R,MAXR,NFRAME)
        CALL PSHDN(RO,MAXR,NFRAME)
        CALL PSHDN(RM,MAXR,NFRAME)
        CALL PSHDN(Q,MAXR,NFRAME)
        CALL PSHDN(U1,MAXR,NFRAME)
        CALL PSHDN(VM,MAXR,NFRAME)
        CALL PSHDN(UO,MAXR,NFRAME)
C
C*  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C*
C*              INSERT ROUTINE TO
C*              "READ IN NFRAME SAMPLES"
        CALL ISAMP(IEOFG9,JFNI,SPEECH,ISCR,N2S,N2E)
C*              CHECK END OF FILE MARKER
        IF(.NOT.IEOFG9) GO TO 2050
C*              END OF FILE MARKER ON
C*              JUMP TO END OF RECEIVER SECTION
2010    NF=NF-1
        GO TO 4050
C*
C*
C*  *  *  *  *  *  *  *  *  *  *  *  *  *  *
C  COMPUTE S/Q RATIO DATA
2050    ESPCH=ENERGY(SPEECH,MAXN,N2S,N2E)
        IZFLG=0
        IF(ESPCH.LE.0.0) GO TO 2100
        IZFLG=-1
        NAPCT=NAPCT+1
        DAPSU=(DAPSU*(NAPCT-1)+DB(ESPCH))/NAPCT
        EAPSU=(EAPSU*(NAPCT-1)+ESPCH)/NAPCT
2100    CONTINUE
C  DO PREEMPHASIS ON INPUT SIGNAL
        IF(.NOT.IPREM) GO TO 2200
        CALL PREEM(SPEECH,S,MAXR,N2S,N2E,PEMFAC)
        GO TO 2300
C  NO PREEMPHASIS - TRANSFER SPEECH TO S
2200    DO 2205 I=N2S,N2E
2205    S(I)=SPEECH(I)
C
```

```
2300    CONTINUE
C
C  PHASED REAL TIME IMPLEMENTATION
C  JUMP FOR RUNG #1
        IF(IRUNG.EQ.1) GO TO 3000
C
C   * * * * * * *  PITCH EXTRACTION  * * * * * * *
C
        IF(.NOT.VPFF) GO TO 2390
        CALL PITCH(S(N2SP),E40,LOG2P,IF0M,IF0L
       1,MPITCH,FR(1),FI(1),PR(1))
C  QUANTIZE PITCH
        IF(.NOT.IQ(6)) GO TO 2305
        IPTCH=MPITCH-IF0M
        MPTCHP=MPITCH
        GO TO 2310
2305    MPTCHP=MPITCH
2310    CONTINUE
C
C   * * * * * * *  PITCH PREDICTION  * * * * * * *
C
C  COMPUTE TAPS
        CALL PTAPS(FR(1),NTAPS,R,MPTCHR,R0)
C  CHECK FOR INSTABILITY OF 3-TAP PREDICTOR
        IF(NTAPS.NE.3) GO TO 2360
        IF(.NOT.INSTPI) GO TO 2360
        CALL STBLIZ(R,NTAPS,R0)
2360    CONTINUE
        IF(.NOT.IQ(7)) GO TO 2312
C  QUANTIZE TAPS
        DO 2311 I=1,NTAPS
        CALL CUDB(R(I),XTAP(1,I),NLAVT(I),IPTAP(I))
        CALL DCUB(IPTAP(I),YTAP(1,I),BR(I))
2311    CONTINUE
        GO TO 2314
C
C  NO QUANTIZATION OF TAPS
2312    DO 2313 I=1,NTAPS
        BR(I)=R(I)
2313    CONTINUE
C
C  PITCH PREDICTION
2314    DO 2315 I=N2S,N2E
        CALL PRED(S,RDP,I,BR,NTAPS,MPTCHR)
C  COMPUTE FIRST RESIDUAL
        R(I)=S(I)+RDP
2315    CONTINUE
        GO TO 2390
C
C  NO PITCH PREDICTION
2390    CONTINUE
        DO 2391 I=N2S,N2E
```



```
2391    R(I)=S(I)
C
2390    CONTINUE
C
C  PHASED REAL-TIME IMPLEMENTATION
C  JUMP FOR RUNG #2
        IF(IRUNG.EQ.2) GO TO 2500
C
C
C  * * * * * * *  LPC ANALYSIS * * * * *
C
        CALL HAMMND(R,MAXR,N2S,N2E,SWR)
        NPWE1=NPWE+1
        CALL AUTCOR(SWD,MAXR,RSCR,NPWE1,N2S,N2E,NPOLE)
C   HIGH FREQUENCY CORRECTION
        IF(.NOT.INFCR) GO TO 2146
        CALL HFCOR(RSCR,NPWFC,CWF,WFLAG)
C  SOLVE LPC NORMAL EQUATIONS TO COMPUTE
C  REFLECTION COEFFICIENTS
2146    CALL ALPCOR(NPOLE,1,RSCR,C,A,V,SCR2)
C  QUANTIZE REFLECTION COEFF
        IF(.NOT.IQ(4)) GO TO 2284
        DO 2150 I=1,NPOLE
        CALL COD(C(I),XCOF(1,I),NLEVC(I),ICQ(I))
        CALL DCOD(ICQ(I),YCOF(1,I),DC(I))
2150    CONTINUE
C  CONVERT REFLECTION COEFF TO PREDICTOR COEFF
        CALL ALPCON(NPOLE,2,RSCR,DC,DA,VP,SCR2)
        GO TO 2288
C  NO QUANTIZATION OF REFLECTION COEFFS
2284    DO 2286 I=1,NPOLE
        CTA(I)=C(I)
        DA(I)=A(I)
2286    DC(I)=C(I)
2288    CONTINUE
C
C  * * * * * * * * *  NOISE SHAPING  * * * * * * * * *
        IF(.NOT.KNSF) GO TO 2151
        CALL PZRS(DA,ARS,PZFAC)
2151    CONTINUE
C
C  * * * * * * *  COMPUTE SECOND RESIDUAL  * * * * * * *
C
        CALL RES(R,MAXR,RD,MAXR,DA,N2S,NPOLE,N2S,N2E)
C
C  PHASED REAL-TIME IMPLEMENTATION
C  JUMP FOR RUNG #3
        IF(IRUNG.EQ.3) GO TO 2500
C
C
C  * * * * * * *  COMPUTE ENERGY AND CODE * * * * * *
```

```
C
2395    CONTINUE
        ENTX=ENERGY(RN,MAXR,N2S,N2E)
        IF(ENTX.EQ.0.) ENTX=ENZERO
        ENRX=SQRT(ENTX)
C  QUANTIZE GAIN
        IF(.NOT.IQ(2)) GO TO 2396
        CALL CUB(ENTX,IG,NLEVG,IENQ)
        CALL DCOB(IENQ,YG,ENRX)
2396    CONTINUE
C  SEGMENTED QUANTIZATION
        IF(.NOT.ISECFG) GO TO 2328
C  COMPUTE DELTA GAINS - ESTX
        ENRSQ=ENRX**ENRX
        CALL DELTAG(RN,MAXR,ENRSQ,N2S,ESTX)
        IF(.NOT.IQ(3)) GO TO 2322
C  QUANTIZE DELTA GAINS
        DO 2321 I=1,NENSEC
        CALL CUB(ESTX(I),ISEC,NLEVSC,IENS(I))
        CALL DCOB(IENS(I),YSEC,ESRX(I))
2321    CONTINUE
        GO TO 2327
C  NO QUANTIZATION OF DELTA GAINS
2322    DO 2326 I=1,NENSEC
        ESRX(I)=SQRT(ESTX(I))
2326    CONTINUE
2327    CONTINUE
C  COMPUTE QUANTIZER SCALE FACTOR - GSEC
        CALL GSCALA(GSAC,ENRX,N2S,ESRX)
        GO TO 2340
C  NO SEGMENTED QUANTIZATION
2328    DO 2330 I=N2S,N2E
2330    GSEC(I)=ENRX
2340    CONTINUE
C
C    * * * * * *  APC LOOP * * * * * *
C
        CALL APC(S,GSEC,ANS,DA,BR,NPTCHR,N2S,CO,IR,YN,RH,Q,Q1
        1,NTAPS,IQ(1),EPPF,RMSF)
C
C
C  FOLDED BINARY CODING
        IF(.NOT.IFBC) GO TO 2289
        CALL FOLBC(IR,NFRAME,IFOLD)
2289    CONTINUE
C
```



```
C  CHANNEL CODING
C  --------------
2400    IF(.NOT.ICHAN) GO TO 2500
C  *  *  ERROR PROTECTION  *  *
C ZERO DATA ARRAY
        DO 2405 I=1,NDATT
2405    IDAT(I)=0
C  CONVERT QUANTIZATION LEVELS OF RESIDUAL SAMPLES
C  IN ARRAY IR(ONE SAMPLE / WORD) INTO A BIT STREAM IB
C  (EACH WORD IN IB REPRESENTS ONE BIT OF THE QUANTIZED RESIDUAL
C   SAMPLES)
        CALL BSTRMI(IR,NOSPA,NFRAME,N2S,IB,NRBITT)
C  LOAD ALL TRANSMITTED PARAMETERS INTO A SINGLE ARRAY IDAT
C  IN THE ORDER GIVEN BY ISEQ. PERCENT OF CHANNEL ERROR FOR
C  EACH PARAMETER,ERP, ALSO LOADED (IN SAME
C  SEQUENCE AS PARAMETERS) IN ARRAY ER.
        CALL SEQTX(NDATT,ISEQ,ICQ,IPTCH,IENQ,IENS,IPTAP,IB,IDAT,ER,ERP)
C  EXTRACT BITS TO BE PROTECTED FROM THE ARRAY IDAT
C  AND FORM 4-BIT BYTES. 4-BIT BYTES ARE PUT IN INW. BITS NOT
C  PROTECTED ARE PUT IN IDATS.
        CALL EXTNHB(NDATP,IDAT,NPER,NBER,NWW,INW,NWNT,IDATS,IRS,ERN,ER)
C  FILL IDATS WITH ALL OTHER PARAMETERS
C  (I.E. ALL PARAMETERS IN WHICH NO BITS WERE PROTECTED)
        DO 2407 I=NDATP+1,NDATT
        IDATS(I)=IDAT(I)
2407    IRS(I)=NBER(I)
C  HAMMING CODE 4-BIT BYTES (INW)
C  INTO 7,4 HAMMING CODEWORDS
        DO 2410 I=1,NWW
        CALL CODH(INW(I),NOPT,INWC(I))
2410    CONTINUE
C
C
C  CHANNEL
C  --------
        IF(.NOT.ICHANE) GO TO 2435
        ICUMNT=0
        IERCUM=0
        NSERR=0
C  ERROR HAMMING CODEWORDS
        DO 2420 I=1,NWW
2420    CALL ERRCHN(INWC(I),NOPT+4,ERN(I),INWE(I),0)
C  ERROR UNPROTECTED DATA BITS
        DO 2430 I=1,NDATT
        KRESFG=0
        IF(I.GE.NSTRES) KRESFG=-1
2430    CALL ERRCHN(IDATS(I),IRS(I),ER(I),IDATE(I),KRESFG)
C  UPDATE COUNT OF TOTAL BITS
        ITCUNB=ITCUNB+ICUMBT
C  UPDATE COUNT OF ERRORED BITS
        ITCUME=ITCUME+IERCUM
```

```
C  UPDATE COUNT OF ERRORED RESIDUAL SAMPLE BITS
        NTSERR=NTSERR+NSERR
        GO TO 2339
C  NO CHANNEL ERROR
2435    CONTINUE
        DO 2436 I=1,NWW
2436    INWL(I)=INWC(I)
        DO 2437 I=1,NDATT
2437    IDATE(I)=IDATS(I)
C
C
C  CHANNEL DECODING
C  ----------------
C  * *  ERROR CORRECTION  * *
C
C  DECODE HAMMING CODEWORDS
2339    DO 2440 I=1,NWW
        CALL DCODH(INWE(I),NUPT,INWR(I),NBADFG)
2440    CONTINUE
C  COMBINE PROTECTED (INWE) AND UNPROTECTED (IDATE) DATA BITS
        CALL CUMBWD(INWR,NPER,NBER,NDATT,IDATE,IDATR)
C  EXTRACT PARAMETERS FROM IDATR
        CALL SEQRX(NDATT,ISEQ,ICQR,IPTCNR,IENQR,IENSR,IPTAPR,IBR,IDATR)
C
C  EXTRACT RESIDUAL SAMPLES FROM BIT STREAM
2400    CALL RSTRMU(IBR,BRUIT,NBSPA,N2S,IRR,NFRAME,NSRCV)
C
        GO TO 3000
C  NO CHANNEL SIMULATION
2500    CONTINUE
        DO 2510 I=1,NPULE
2510    ICQR(I)=ICQ(I)
        DO 2520 I=1,NFRAME
2520    IRR(I)=IR(I)
        DO 2530 I=1,NENSEC
2530    IENSR(I)=IENS(I)
        DO 2540 I=1,NTAPS
2540    IPTAPR(I)=IPTAP(I)
        IENQR=IENQ
        IPTCNR=IPTCN
C
^L
```



```
C  *  *  *  *  *  RECEIVER  *  *  *  *  *
3000    CONTINUE
C
C  PUSH DOWN ARRAYS
        CALL PSHDN(RR,MAXR,NFRAME)
        CALL PSHDN(RRH,MAXR,NFRAME)
        CALL PSHDN(NMES,MAXR,NFRAME)
        CALL PSHDN(SP,MAXR,NFRAME)
        CALL PSHDN(SLAST,MAXR,NFRAME)
C
C
C  RUNG BRANCHING
        IF(IRUNG.NE.1) GO TO 3004
        DO 3002 I=N2S,N2E
        SLAST(I)=S(I)
3002    CONTINUE
        GO TO 4000
C
3004    IF(IRUNG.NE.2) GO TO 3008
        DO 3006 I=N2S,N2E
        SP(I)=R(I)
3006    CONTINUE
        GO TO 3510
C
3008    IF(IRUNG.NE.3) GO TO 3012
        DO 3010 I=N2S,N2E
        RWW(I)=RW(I)
3010    CONTINUE
        GO TO 3410
C
3012    CONTINUE
C
C  * * * * * * * * * * *  DECODE ENERGY  * * * * * * *
        IF(IQ(2)) GO TO 3110
C  NO QUANTIZATION
        ENRX=SQRT(ENTX)
        GO TO 3112
C  QUANTIZED ENERGY
3110    CALL DCOD(IENQR,VG,ENRX)
3112    IF(ISECPG) GO TO 3116
C  NO SEGMENTED QUANTIZATION
        DO 3114 I=N2S,N2E
3114    GSEC(I)=ENRX
        GO TO 3135
C  SEGMENTED QUANTIZATION
3116    IF(.NOT.IQ(3)) GO TO 3125
C  QUANTIZED DELTA GAINS
        DO 3118 I=1,NENSEC
        CALL DCOD(IENSR(I),VSEC,ESRX(I))
3118    CONTINUE
        GO TO 3130
```

```
C  RU QUANTIZATION OF DELTA GAINS
3125     DO 3127 I=1,NLRSEC
         RSRA(I)=SQRT(ESTR(I))
3127     CONTINUE
3130     CALL QSCALE(GSEC,ERRR,N2S,RSRX)
3135     CONTINUE
C
C* * * * * **   DECODE RESIDUAL   * * * * * *
C
C  DECODING FOR FOLDED BINARY CODING
         IF(.NOT.IFBC) GO TO 3210
         CALL FOLDC(IRR,NFRAME,IFOLD)
3210     CONTINUE
C
C  DECODE RESIDUAL SAMPLES
         IF(.NOT.IQ(1)) GO TO 3220
         K=N2S-1
         DO 3214 I=1,NFRAME
         K=K+1
         CALL DCOD(IRR(I),RMS>,RR(K))
3214     CONTINUE
         GO TO 3230
C
C  NO QUANTIZATION - TRANSFER GO TO RR
3220     DO 3225 I=N2S,N2E
3225     RR(I)=RD(I)
3230     CONTINUE
C
C   * * * * * * *   UPDATE  ENERGY  * * * * * * *
C
         DO 3310 I=N2S,N2E
         RRR(I)=RR(I)*GSEC(I)
3310     CONTINUE
C
C    * * * * * * * * *   DECODE SPECTRAL COEFFS  * * * * * * * * *
C
3410     CONTINUE
         IF(IQ(4)) GO TO 3430
C  NO QUANTIZATION
         DO 3420 I=1,NPOLE
3420     CR(I)=CTR(I)
         GO TO 3450
C  QUANTIZATION OF SPECTRAL COEFF
3430     DO 3440 I=1,NPOLE
         CALL DCOD(ICWR(I),VCWF(1,I),CR(I))
3440     CONTINUE
C  COMPUTE PREDICTOR COEFF
3450     CONTINUE
         CALL ALPCON(NPOLE,2,*SCR,CR,AR,V,SCR2)
C
C   *   *   *   *   *  SPECTRAL SYNTHESIS   *    *    *   *
```



```
C
         CALL SYNTH(SP,MAXR,AR,RRR,MAXR,N2S,NPOLE,N2S,N2E)
C
C
C    TRANSFER SP TO SLAST IF NO PITCH PREDICTION
         IF(KPPF) GO TO 3510
         DO 3460 I=N2S,N2E
         SLAST(I)=SP(I)
3460     CONTINUE
         GO TO 4000
C
C  * * * * * * * * *  DECODE PITCH   * * * ** * * * *
C
3510     CONTINUE
         IF(.NOT.IQ(6)) GO TO 3515
         MPTCHR=IPTCHR+IFON
         GO TO 3517
3515     MPTCHR=MPITCH
3517     CONTINUE
C
C  * * * * * * * * DECODE PITCH TAPS   * * * * * * * * *
C
         IF(.NOT.IQ(7)) GO TO 3525
C  QUANTIZATION ON
         DO 3520 I=1,NTAPS
         CALL DCOD(IPTAPR(I),YTAP(1,I),BR(I))
3520     CONTINUE
         GO TO 3530
C   NO PITCH TAP QUANTIZATION
3525     DO 3527 I=1,NTAPS
3527     BR(I)=B(I)
3530     CONTINUE
C
C   *  *  *  *  *  *   PITCH SYNTHESIS  *   *   *   *   *
C
C  SAVE SP FOR NEXT FRAME
         DO 3535 I=N2S,N2E
3535     RRES(I)=SP(I)
         DO 3538 I=N2S,N2E
         RRRR=0.
C  COMPUTE LINEARLY PREDICTED OUTPUT VALUE
C  USING DECODED PITCH PREDICTOR FILTER.
         CALL PREV(RRES,RRRR,I,BR,NTAPS,MPTCHR)
         RRES(I)=RRES(I)-RRRR
3538     CONTINUE
C  TRANSFER TO SLAST
         DO 3542 I=N2S,N2E
3542     SLAST(I)=RRES(I)
C
C  DEEMPHASIS OF OUTPUT
4000     IF(IPREM)CALL DEMPH(SLAST,MAXR,N2S,N2E,PEMFAC)
```

```
C* * * * * * * * * * * * * * *
C*
C*              INSERT ROUTINE TO
C*              "OUTPUT SAMPLES"
      CALL USAMP(JFNO,SLAST,ISCR,N2S,N2E)
C*
C*
C* * * * * * * * * * * * * * *
C  COMPUTE OUTPUT QUANTIZATION NOISE (SLAST-SPEECH)
C
      DO 4010 I=N2S,N2E
4010  SPQ(I)=SLAST(I)-SPEECH(I)
C
C  COMPUTE RUNNING AVERAGE OF QUANTIZATION  NOISE
      EAPQ=ENERGY(SPQ,WAIR,N2S,N2E)
      IF(.NOT.IZFLG) GO TO 4012
      EAPQU=(EAPQO*(NAPCT-1)+DB(EAPQ))/NAPCT
      EAPQO=(EAPQU*(NAPCT-1)+EAPQ)/NAPCT
4012  CONTINUE
C  RETURN TO TOP OF TRANSMITTER SECTION  FOR PROCESSING NEXT FRAME
      GO TO 2000
C* * * * * * * * * * * * * * *
C*
C*              INSERT ROUTINE TO
C*              "CLOSE FILES"
C*              UTTERANCE FINISHED
4050  CALL CLOSF(JFNU)
      CALL CLOSF(JFNI)
      JFNI=-1
      JFNO=-1
C
C* * * * * * * * * * * * * * *
C
C  COMPUTE PERCENT CHANNEL ERROR AND PRINT
      IF(.NOT.ICHAN) GO TO 4110
      IF(ITCUMB.EQ.0) GO TO 4067
      PERERR=(FLOAT(ITCUME)/FLOAT(ITCUMB))*100.
      PERRES=(FLOAT(NTSERR)/FLOAT(NF*NRES(T)))*100.
      GO TO 4069
4067  PERERR=0.0
      PERRES=0.0
4069  TYPE 4072,PERERR
4072  FORMAT(1H ,14X,17HCOMPUTED % ERROR=,F9.4,/)
      TYPE 4085,PERRES
4085  FORMAT(1H ,14X,22HCOMPUTED % ERROR(RES)=,F9.4,/)
4110  CONTINUE
C
C  COMPUTE S/Q RATIOS AND PRINT
      DBAS=DB(EAPSU)
      DBSQ=0.
      IF(EAPQO.GT.0.) DBSQ=DB(EAPQU)
      TYPE 4222,NAPCT
```



```
4222  FORMAT(1H ,14X,12HFRAME COUNT=,I3)
      SSQO=DAPSO-DAPQO
      ASQO=DBES-DBSQ
      TYPE 4240,ASQO,SSQO
4240  FORMAT(1H ,14X,27HS/Q RATIO IN DB: LONG-TERM=,F8.3
     1,12H) SEGMENTAL=,F8.3)
4250  CONTINUE
C
C  CONTINUE
      TYPE 10010
10010 FORMAT(1H ,22HCONTINUE(YES=-1,NO=0)=,$)
      ACCEPT 10000,ICONT
10000 FORMAT(I)
      IF(.NOT.ICONT) GO TO 10020
C
C  RETURN TO TOP OF PROGRAM FOR PROCESSING NEXT UTTERANCE FILE
      GO TO 100
10020 STOP
      END
```

```
C  <DCA16>16AIO.F4.2   Tue 8-Jul-80 11:50AM          Page 1

      SUBROUTINE ISAMP(IFLG,JFNI,S,ISCR,NST,NEN)

C  SUBROUTINE TO INPUT SAMPLES FROM FILE
C    INPUT:
C       JFNI = JFN (JOB FILE NUMBER) OF INPUT FILE
C    OUTPUTS:
C          S  = VECTOR OF SAMPLES (REAL)
C               SAMPLES STORED IN S AS 36-BIT FLOATING- POINT NUMBERS
C               FROM NST TO NEN
C       ISCR = SCRATCH VECTOR (SAME SIZE AS S)
C       IFLG = IFLG IS SET (IFLG=-1) IF END OF FILE
C              IS ENCOUNTERED WHEN READING IN SAMPLES

      COMMON/WAVFIL/IWBSIZ,IEXWD,IWDYTI
      COMMON/FRMSIZ/NFRAME,NWIND
      COMMON/INEFPT/INEFPT

      DIMENSION S(1),ISCR(1)

      IFLG=0
      ASSIGN 100 TO IERR
C   SET FILE POINTER TO CORRESPOND TO THE
C   BEGINNING OF THE DESIRED FRAME.
      CALL SPTR(JFNI,IWDYTI*(INEFPT)
C   READ NWIND NUMBER OF INTEGER SAMPLES OF SIZE
C   IWBSIZ BITS EACH, INTO THE VECTOR ISCR(SAMPLES ARE RIGHT JUSTIFIED)
      CALL SINB(IERR,JFNI,ISCR(NST),36,NWIND)
C   FOR EACH SAMPLE, EXTEND SIGN BIT INTO
C   (36-IWBSIZ) MOST SIGNIFICANT BITS, AND CONVERT
C   TO FLOATING- POINT NUMBER.
      DO 10 I=NST,NEN
      S(I)=EXTFLT(ISCR(I),IEXWD)
10    CONTINUE
C   ADVANCE THE COUNTER TO CORRESPOND
C   TO THE BEGINNING OF THE NEXT FRAME.
      INEFPT=INEFPT+NFRAME
      RETURN
C  SET END OF FILE FLAG
100   IFLG=-1
      RETURN
      END
^L
```

```
C  <DCA16>16AIO.F4.2   Tue 8-Jul-80 11:50AM          Page 2

      SUBROUTINE OSAMP(JFNO,S,ISCR,NST,NEN)
C
C  SUBROUTINE TO OUTPUT SAMPLES TO FILE
C    INPUTS:
C       JFNO = JFN OF OUTPUT FILE
C          S  = VECTOR OF SAMPLES (REAL)
C               SAMPLES IN S FROM NST TO NEN ARE TRANSFERED TO FILE
C       ISCR = SCRATCH VECTOR (SAME SIZE AS S)
C
      DIMENSION S(1),ISCR(1)
      COMMON/MAXAMP/MAXAMP,NOVER
C  NOVER IS THE NUMBER OF TIMES SAMPLE AMPLITUDES
C  EXCEEDS AND HENCE SATURATES AT MAXAMP.
C
      K=0
      DO 10 I=NST,NEN
      K=K+1
C  FIX AND ROUND THE SAMPLES
      ISCR(K)=IFIXR(S(I))
C   SATURATE SAMPLE AMPLITUDE AT MAXAMP
      IF(ISCR(K).GE.MAXAMP) GO TO 20
      GO TO 30
20    ISCR(K)=MAXAMP-1
      NOVER=NOVER+1
30    IF(ISCR(K).LT.-MAXAMP) GO TO 40
      GO TO 10
40    ISCR(K)=-MAXAMP
      NOVER=NOVER+1
10    CONTINUE
C  TRANSFER K SAMPLES IN ISCR
C  TO OUTPUT FILE
      CALL SOUTB(JFNO,ISCR,36,K)
      RETURN
      END
^L
```

```
      SUBROUTINE OPNIF(IERR,JFNI)
C
C  SUBROUTINE TO OPEN INPUT FILE
C     OUTPUT:
C           JFNI = JFN OF INPUT FILE
C                  JFNI IS GT 0 IF FILE
C                  WAS SUCESSFULLY OPENED.
C                  IF UNSUCCESSFULL,
C                  JFNI=-1
C           IERR = FLAG TO INDICATE TROUBLE
C                  WITH FILE OPENING
C
      COMMON/WAVFIL/INBSIZ,IEXWD,INBYTI
      COMMON/NAME/TMPNAM(10),INAME(10),ONAME(10)
      COMMON/IHDR/IHDR(30),TESHDR
      COMMON/FREQS/FREQS,TSAMP
      COMMON/MAXAMP/MAXAMP,NOVER
      COMMON/IREFPT/IREFPT
      COMMON/NFLPTS/NFLPTS
      INTEGER NBRTES(30)
      DATA (NUMTES(I),I=1,30)/30,10,1,0,0,1,1,23**4000000000000/
C
      IERR=0
300   CALL PSOUTR
       CALL PSOUT('INPUT SPEECH FILE: ')
      CALL RALPN(INAME,NCH)
      IF(NCH.EQ.0) GO TO 302
      GO TO 310
302   CALL PSOUTR('? NAME NOT GIVEN - RESTART')
      IERR=-1
      RETURN
C
C  FILE NAME NOT EMPTY - CHECK IF EXTENSION IS SPECIFIED
310   NBTNM=NBYTS(INAME)
      DO 312 I=1,NBTNM
      IF(ICHAR(INAME,I).EQ.'.') GO TO 314
312   CONTINUE
C  NO EXTENSION GIVEN
      CALL PSOUTR('NO EXTENSION GIVEN - RESTART')
      IERR=-1
      RETURN
C   EXTENSION GIVEN - STORE <DIR>NAME IN TMPNAM
314   CALL CHMOVE(INAME,1,TMPNAM,1,I-1)
C  IF FILE IS OPEN (JFNI IS NOT NEGATIVE)
C  CLOSE THE FILE WITH JFN=JFNI
      IF(JFNI.GT.0) CALL CLOSF(JFNI)
      JFNI=-1
C  OPEN FILE
      CALL OPENIF(JFNI,36,INAME,IER)
      IF(IER.EQ.0) GO TO 320
      CALL PSOUTR('TROUBLE WITH OPENIF - RESTART')
      JFNI=-1
      IERR=-1
      RETURN
```



```
C
C  READ IN 1ST 30 WORDS FROM FILE  AND FIND OUT IF HEADER EXISTS
320   ASSIGN 321 TO IEOF
      CALL SIN#(IEOF,JFNI,IHDR,36,30)
      GO TO 322
C  IEOF CALL
321   CALL PSOUTR('UNEXPECTED IEOF ON HEADER READ - RESTART')
      IERR=-1
      RETURN
322   IF(IHDR(1).EQ.NBRTES(1)
      1.AND.IHDR(2).EQ.NBRTES(2)
      2.AND.IHDR(3).EQ.NBRTES(3)
      4.AND.IHDR(6).EQ.NBRTES(6)
      5.AND.IHDR(7).EQ.NBRTES(7)) GO TO 332
C  FILE HAS NO HEADER
C  ASK FOR FILE BYTE SIZE AND SAMPLING RATE
      TESHDR=0.
      CALL PSOUTR(' FILE DOES NOT HAVE HEADER')
      CALL PSOUT(' FILE SAMPLING RATE (HZ): &')
      ACCEPT 101,FREQS
101   FORMAT(F)
      TSAMP=1./FREQS
      CALL PSOUT(' FILE BYTE SIZE(IN BITS): &')
      ACCEPT 102,INBSIZ
102   FORMAT(I)
      IEXWD=LSH(1,INBSIZ-1)
      INBYTI=0
      MAXAMP=2**(INBSIZ-1)
      NOVER=0
C  FIND FILE LENGTH NFLPTS IN TERMS OF BYTE SIZE JUST GIVEN
      CALL SFBSZ(JFNI,INBSIZ)
      CALL SFPTR(JFNI,-1)
      CALL RFPTR(JFNI,NFLPTS)
      CALL SFPTR(JFNI,0)
C  MAKE HEADER FOR OUTPUT FILE
      DO 330 I=1,30
330   IHDR(I)=NBRTES(I)
      IHDR(4)=INBSIZ
      SMP=(1./FREQS)*1.0E+6
      IHDR(11)=IFIXR(SMP/10.)
      TMP=10.*FLOAT(IHDR(11))
      IF(SMP.NE.TMP) IHDR(11)=-IFIXR(SMP)
      IREFPT=0
      GO TO 335
C
C  FILE HAS HEADER - USE HEADER DATA
```

```
332     FESHDR=-1.
        CALL SFBSZ(JFN1,IHDR(4))
        IWBSIZ=IHDR(4)
        NBYTWD=36/IWBSIZ
        IEXWD=LSH(1,IWBSIZ-1)
        IWBYT1=IHDR(1)*NBYTWD
        MAXAMP=2**(IWBSIZ-1)
        NOVER=0
        NFLPTS=IHDR(5)
        IF(IHDR(11).LT.0) GO TO 333
        FREQS=(1./FLOAT((HDR(11))))*.1E+6
        GO TO 334
333     FREQS=1.0E+6/(FLOAT((ABS((HDR(11)))))
334     TSAMP=1./FREQS
        IREPPT=0
C
C  COMPUTE UTTERANCE LENGTH IN SECONDS
C  AND TYPE IT OUT AT THE USER'S TERMINAL
335     II=NFLPTS*1000./FREQS+0.5
        JJ=TSAMP*1E+6
        UUR=II/1000.
        TYPE 337,NFLPTS,IWBSIZ,JJ,UUR
337     FORMAT('+',I7,I3,'-BIT SAMPLES AT',I4,' USEC =',
        1F7.4,' SECONDS',/)
C
        RETURN
        END
```



```
        SUBROUTINE OPNOF(IERR,JFNO)
C  SUBROUTINE TO OPEN OUTPUT SPEECH FILE
C       OUTPUT:
C               JFNO = JFN OF OUTPUT FILE
C                      (AS IN OPNIF)
C               IERR = FLAG TO INDICATE TROUBLE
C                      WITH FILE OPENING
C
        COMMON/WAVFIL/IWBSIZ,IEXWD,IWBYT1
        COMMON/IHDR/IHDR(30),FESHDR
        COMMON/NAME/TMPNAM(10),INAME(10),ONAME(10)
C
        IERR=0
15      CALL PSOUT('OUTPUT SPEECH FILE:')
        CALL OPENOF(JFNO,36,0,IER)
        IF(IER.EQ.0) GO TO 20
        CALL PSOUTR('TROUBLE WITH OPENOF - RESTART')
        IERR=-1
        RETURN
C  STORE NAME
20      DO 22 I=1,10
22      ONAME(I)=0.
        CALL FILNAM(JFNO,ONAME)
        CALL SFBSZ(JFNO,36)
C  OUTPUT HEADER AND SET BYTE SIZE
        CALL SOUTB(JFNO,IHDR,36,30)
        CALL SFBSZ(JFNO,IWBSIZ)
        RETURN
        END
```

```
      FUNCTION ALAR(C)
C
C  REFLECTION COEFF TO LOG AREA RATIO
C  CONVERSION FUNCTION
C
      CXX=(1.+C)/(1.-C)
      ALAR=DB(CXX)
      RETURN
      END




      FUNCTION REFLN(A)
C
C  LOG AREA RATIO TO REFLECTION COEFF
C  CONVERSION FUNCTION
C
      YY=(10.0)**(A/10.0)
      REFLN=(YY-1.)/(1.+YY)
      RETURN
      END




      FUNCTION DB(X)
C
C  DB FUNCTION: 10*LOG10(X)
C
      XX=X
      IF(XX.LT.1.0E-10) XX=1.0E-10
      DB=10.*ALOG10(XX)
      RETURN
      END




      FUNCTION RDB(A)
C
C  "REVERSE" DB FUNCTION: 10**(A/10)
C
      B=A/10.0
      RDB=(10.0)**B
      RETURN
      END
```



```
      FUNCTION IFIXR(A)
C
C  FIX AND ROUND A FLOATING
C  POINT NUMBER
C
      AB=ABS(A)
      IF(AB.NE.0.) GO TO 10
      IFIXR=0
      RETURN
10    ASIGN=A/AB
      TEMP=A+ASIGN*(0.5)
      IFIXR=IFIX(TEMP)
      RETURN
      END
C




      SUBROUTINE PSHDN(A,ND,N)
C
C  ROUTINE TO PUSH DOWN (PSHDN) DATA IN AN
C  ARRAY (A) OF TOTAL LENGTH ND BY AN AMOUNT N
C  THAT IS , THE CONTENTS OF A(N+1) THROUGH
C  A(ND) ARE COPIED INTO A(1) THROUGH A(ND-N);
C  THE CONTENTS OF A(ND-N+1) THROUGH A(ND) ARE
C  UNCHANGED.
C
      DIMENSION A(ND)
      N1=N+1
      K=0
      DO 10 I=N1,ND
      K=K+1
10    A(K)=A(I)
      RETURN
      END




      SUBROUTINE SETZR(X,MAXX)
C
C  ROUTINE TO ZERO OUT AN ARRAY (X)
C  OF LENGTH MAXX
C
      DIMENSION X(1)
      DO 10 I=1,MAXX
      X(I)=0.0
10    CONTINUE
      RETURN
      END
```

```
C  <DCA16>16KCUD.F4.2   Thu 21-Aug-80 9:37AM          Page 1

      SUBROUTINE QTBL(BMX,BMN,NLEV,X,Y,STEP)
C
C  COMPUTES QUANTIZATION TABLE VALUES, FOR UNIFORM QUANTIZER
C    INPUT:
C       BMX  = MAXIMUM QUANTIZER RANGE
C       BMN  = MINIMUM QUANTIZER RANGE
C       NLEV = NUMBER OF QUANTIZATION LEVELS
C    OUTPUT:
C       X    = VECTOR OF QUANTIZER THRESHOLDS (TRANSMITTER CODING TABLE)
C       Y    = VECTOR OF QUANTIZED PARAMETER VALUES (RCVR DECODING
C                TABLE)
C       STEP = STEP SIZE
C
C
      DIMENSION X(1),Y(1)
C
      RNG=BMX-BMN
      STEP=RNG/FLOAT(NLEV)
      X(1)=BMN
      STEP2=STEP*.5
      DO 10 I=1,NLEV
      X(I+1)=X(I)+STEP
      Y(I)=X(I)+STEP2
10    CONTINUE
      RETURN
      END
^L
```

```
C  <DCA16>16KCUD.F4.2   Thu 21-Aug-80 9:37AM          Page 2

      SUBROUTINE QTBLT(NTAPS,NBITT,NLEVT,XTAP,YTAP,TSTEP)
C
C  COMPUTES CODING TABLES FOR PITCH PREDICTOR TAPS
C  (NOTE : ONLY ONE OR THREE TAP CASES PERMITTED)
C
C  INPUTS:
C       NTAPS   NUMBER OF PITCH PREDICTOR TAPS
C       NBITT   VECTOR OF THE NUMBER OF BITS USED TO CODE TAPS
C  OUTPUTS:
C       NLEVT   VECTOR OF THE NUMBER OF QUANTIZATION
C               LEVELS FOR EACH TAP
C       X       VECTOR OF QUANTIZER THRESHOLDS(TRANSMITTER CODING TABLE)
C       Y       VECTOR OF QUANTIZED PARAMETER VALUES
C               (RECEIVER DECODING TABLE)
C       TSTEP   STEP SIZE
C
C
C
C
      DIMENSION NBITT(1),NLEVT(1),XTAP(33,3),YTAP(32,3)
      DIMENSION TSTEP(1),TMAX0(3),TMIN0(3),TMAX(3),TMIN(3)
C
      DATA TMIN0(1)/-0.549/,TMIN0(2)/-0.95/,TMIN0(3)/-0.549/
      DATA TMAX0(1)/0.427/,TMAX0(2)/0.12/,TMAX0(3)/0.427/
C
      IF(NTAPS.EQ.1) GO TO 1200
C  3-TAP CASE
      DO 110 I=1,NTAPS
      TMAX(I)=TMAX0(I)
      TMIN(I)=TMIN0(I)
110   CONTINUE
      GO TO 125
C
C  ONE TAP CASE
1200  TMAX(1)=TMAX0(2)
      TMIN(1)=TMIN0(2)
C
125   CONTINUE
      DO 127 I=1,NTAPS
      NLEVT(I)=2**NBITT(I)
127   CONTINUE
      DO 129 I=1,NTAPS
      CALL QTBL(TMAX(I),TMIN(I),NLEVT(I),XTAP(1,I),YTAP(1,I),TSTEP(I))
129   CONTINUE
      RETURN
      END
^L
```

```
      SUBROUTINE QTBLC(NPOLE,NBITC,NLEVC,XCOF,YCOF,CSTEP)
C
C  ROUTINE TO SET CODING TABLES FOR REFLECTION COEFFICIENTS.
C    INPUT:
C       NPOLE = NUMBER OF COEFFICIENTS
C       NBITC = VECTOR OF NUMBER OF BITS PER COEFFICIENT
C    OUTPUT:
C       NLEVC = VECTOR OF NUMBER OF LEVELS PER COEFFICIENT
C       XCOF  = TRANSMITTER CODING TABLE
C       YCOF  = RECEIVER DECODING TABLE
C       CSTEP = VECTOR OF STEP SIZES
C
C
      DIMENSION NBITC(1),NLEVC(1),XCOF(129,14),YCOF(128,14)
      DIMENSION CSTEP(1),CMIN(14),CMAX(14)
C
C  THE FOLOLOWING DATA STATEMENTS CONTAIN LAR CODING RANGES IN DB:
      DATA CMIN(1)/-21.849249/,CMIN(2)/-0.7087499/,CMIN(3)/-9.83125/
      DATA CMIN(4)/-6.89375/,CMIN(5)/-5.28125/,CMIN(6)/-3.74805/
      DATA CMAX(1)/11.853149/,CMAX(2)/13.710449/,CMAX(3)/7.96875/
      DATA CMAX(4)/8.90625/,CMAX(5)/7.71875/,CMAX(6)/9.5599499/
C
      DO 110 I=1,NPOLE
      NLEVC(I)=2**NBITC(I)
110   CONTINUE
C
      DO 111 I=1,NPOLE
      CALL QTBL(CMAX(I),CMIN(I),NLEVC(I),XCOF(1,I),YCOF(1,I),CSTEP(I))
111   CONTINUE
C
100   CONTINUE
C
C  CONVERT TABLE VALUES FOR DIRECT CODING
C  OF REFLN COEFF
      DO 112 I=1,NPOLE
      N=NLEVC(I)
      XCOF(1,I)=REFLN(XCOF(1,I))
      DO 112 J=1,N
      XCOF(J+1,I)=REFLN(XCOF(J+1,I))
      YCOF(J,I)=REFLN(YCOF(J,I))
112   CONTINUE
      RETURN
      END
^L
```



```
      SUBROUTINE QTBLG(NBITG,NLEVG,XG,YG,GSTEP)
C
C
C  ROUTINE TO SET UP CODING TABLES FOR OVERALL ENERGY
C
C  INPUTS:
C       NBITG    NUMBER OF BITS FOR CODING ENERGY
C  OUTPUTS:
C       NLEVG    NUMBER OF QUANTIZATION LEVELS
C       XG       TRANSMITTER CODING TABLE
C       YG       RECEIVER DECODING TABLE
C       GSTEP    STEP SIZE
C
C
      DIMENSION XG(1),YG(1)
C
C  CMAX AND CMIN DEFINE GAIN CODING RANGE IN DB:
C   NOTE: THE VALUES CMAX AND CMIN GIVEN BELOW HAVE
C   BEEN EMPIRICALLY OBTAINED FOR INPUT SPEECH
C   SAMPLES THAT HAVE BEEN NORMALIZED(OVER INDIVIDUAL
C   UTTERANCES) TO BE EACH 11 BITS (INCLUDING ONE SIGN BIT).
C   FOR OTHER SPEECH SAMPLE BYTE SIZES , FIGURE
C   APPROXIMATELY 6 DB/BIT CHANGE ON THE VALUES OF
C  CMAX AND CMIN. THIS IS ACCOMPLISHED BY SETTING
C  THE VALUE OF DBCHANG TO BE EQUAL TO THE TOTAL CHANGE IN
C  DB; THIS CHANGE IS POSITIVE IF THE SPEECH BYTE SIZE IS
C  GREATER THAN 11 BITS AND NEGATIVE IF IT IS LESS.
C  FOR EXAMPLE , IF THE BYTE SIZE IS 9, DBCHANG=-12.0
      DATA CMAX/46.0/,CMIN/-18.0/
      DATA DBCHANG/0.0/
C
      NLEVG=2**NBITG
      CMAXS=CMAX+DBCHANG
      CMINS=CMIN+DBCHANG
C
      CALL QTBL(CMAXS,CMINS,NLEVG,XG,YG,GSTEP)
C
C  CONVERT TABLE VALUES FOR DIRECT CODING
C  OF ENERGY
      XG(1)=RDB(XG(1))
      DO 115 I=1,NLEVG
      XG(I+1)=RDB(XG(I+1))
      YG(I)=SQRT(RDB(YG(I)))
115   CONTINUE
      RETURN
      END
^L
```

```
      SUBROUTINE QTBLSC(NBITSC,NLEVSC,XSEC,YSEC)
C
C  ROUTINE TO SET UP CODING TABLES FOR DELTA GAINS.
C
C  INPUT:
C       NBITSC  NUMBER OF BITS FOR CODING DELTA GAINS
C               ( SET UP FOR ONLY NBITSC = 2 BITS)
C  OUTPUTS:
C       NLEVSC  NUMBER OF QUANTIZATION LEVELS
C       XSEC    TRANSMITTER CODING TABLE
C       YSEC    RECEIVER DECODING TABLE
C
C
      DIMENSION XSEC(1),YSEC(1)
      DIMENSION XSEC0(5),YSEC0(4)
C
C  THE FOLLOWING TABLE VALUES ARE IN DB:
      DATA XSEC0(1)/-15.0/,XSEC0(2)/-3.6/,XSEC0(3)/-0.5/
      DATA XSEC0(4)/2.2/,XSEC0(5)/30.0/
      DATA YSEC0(1)/-6.2/,YSEC0(2)/-2.0/,YSEC0(3)/1.0/
      DATA YSEC0(4)/3.5/
C
      NLEVSC=2**NBITSC
C
C  CONVERT TABLE VALUES FOR DIRECT CODING
C  OF ENERGY RATIOS
      XSEC(1)=RDB(XSEC0(1))
      DO 100 I=1,NLEVSC
      XSEC(I+1)=RDB(XSEC0(I+1))
      YSEC(I)=SQRT(RDB(YSEC0(I)))
100   CONTINUE
      RETURN
      END
^L
```



```
      SUBROUTINE COD(R,X,NLEV,IR)
C
C   GENERAL CODING ROUTINE (TABLE LOOKUP)
C   INPUT:
C       R    = PARAMETER TO BE CODED
C       X    = TRANSMITTER CODING TABLE ( QUANTIZER THRESHOLDS)
C       NLEV= # OF QUANTIZATION LEVELS
C   OUTPUT:
C       IR = TRANSMITTED CODE
C
      DIMENSION X(NLEV)
C
      DO 10 N=2,NLEV
      IF(R.LT.X(N)) GO TO 100
10    CONTINUE
      N=NLEV+1
100   IR=N-2
      RETURN
      END
^L
```

```
      SUBROUTINE DCOD(IR,V,R)
C
C  GENERAL DECODING ROUTINE (TABLE LOOKUP)
C    INPUT:
C       IR = RECEIVED CODE FOR PARAMETER R
C       V  = RECEIVER DECODING TABLE
C    OUTPUT:
C       R  = QUANTIZED VALUE OF R
C
C
C
      DIMENSION V(1)
C
      R=V(IR+1)
      RETURN
      END
```



```
C
C
      SUBROUTINE OPTQ(ITYP,NLEVS,X,V)
C
C  PROGRAM TO SET UP CODING AND DECODING TABLES FOR OPTIMUM
C  NONUNIFORM QUANTIZATION.  IT IS ASSUMED THAT THE INPUT TO
C  THE QUANTIZER HAS ZERO MEAN AND UNIT VARIANCE.  THE
C  ALGORITHM IS GIVEN IN MAX, "QUANTIZING FOR MINIMUM DISTORTION,"
C  IRE TRANS. INFORM. THEORY, VOL IT-6, PP.7-12, MAR 1960.
C
C    INPUTS:
C       ITYP    MODEL P.D.F.     1=GAMMA
C                                2=LAPLACIAN
C                                3=GAUSSIAN
C       NLEVS   NUMBER OF QUANTIZATION LEVELS.
C    OUTPUTS:
C       X       CODING TABLE.
C       V       DECODING TABLE.  X(I) < V(I) < X(I+1).
C
      DIMENSION P(2000),X(1),V(1)
      DATA BINWID/0.01/,EPS/1.0E-10/,TOL/0.002/
C
C  EVALUATE MODEL DISTRIBUTION FUNCTION, F(X), AT INCREMENTS
C  OF BINWID TO GENERATE A ONE-SIDED HISTOGRAM, P.
C  P(I)= PROBABILITY THAT (I-1)*BINWID < ABS(X) < I*BINWID.
      TOT=0.
      SIGMA=0.
      P2= 0.5
      I=0
70    I= I+1
      Z= FLOAT(I)*BINWID
      P1=P2
      IF(ITYP.NE.1) GO TO 85
C  NDTR (FROM IBM SSP) EVALUATES GAUSSIAN ERROR FUNCTION.
80    Z= SQRT(2.*Z)
      CALL NDTR(Z,P2,D)
85    IF(ITYP.EQ.2) P2= 1.-.5*EXP(-1.4142*Z)
      IF(ITYP.EQ.3) CALL NDTR(Z,P2,D)
      DIF= P2 -P1
C  QUIT WHEN HISTOGRAM APPROACHES ZERO.
      IF(DIF.LT.EPS) GO TO 78
      P(I)= DIF
      TOT= TOT+DIF
      XX= FLOAT(I-1)
      SIGMA= SIGMA +P(I)*(XX*XX +XX +.333)
      GO TO 75
78    SIGMA= SQRT(SIGMA/TOT)
      MAX= I-1
C
C  DETERMINE THE OPTIMAL (M.M.S.E.) QUANTIZER FOR THE GIVEN
C  HISTOGRAM.  START WITH GUESS OF INNERMOST DECODING LEVEL.
      GUESS= SIGMA/FLOAT(NLEVS)
```

```
C  EVEN OR ODD NUMBER OF LEVELS.
      M= MOD(NLEVS,2)
      NLEV2= NLEVS/2
      IF(M.EQ.0) X(1)= 0.
      IF(M.EQ.1) Y(1)= 0.
C  ITERATE UP TO 100 TIMES
      DO 10 ITT=1,100
40    Y(1+M)= GUESS
      IF(M.EQ.1) X(1)= Y(2)/2.
C  CONSTRUCT QUANTIZER RECURSIVELY
      DO 50 I=1,NLEV2-1
      CALL XBEST(P,MAX,X(I),Y(I+M),X(I+1),IFLAG)
      IF(.NOT.IFLAG) GO TO 30
20    GUESS= GUESS*0.9
      GO TO 40
30    Y(I+1+M)= 2.*X(I+1) -Y(I+M)
50    CONTINUE
      XMP1= FLOAT(MAX)-EPS
      X(NLEV2+1)= XMP1
C  CHECK CENTROID OF LAST BIN AND ADJUST GUESS.
      CALL CENTRD(P,MAX,X(NLEV2),TEST,XMP1)
      COR= (TEST-Y(NLEV2+M))/Y(NLEV2+M)
      IF(ABS(COR).LT.TOL) GO TO 95
10    GUESS= Y(1+M) +Y(1+M)*COR*0.75
C
C  SCALE BY STD.DEV. AND MAKE SYMMETRIC W.R.T. 0.
95    IF(M.EQ.1) TEMP= Y(NLEV2+1)
      DO 90 I=1,NLEV2+M
      X(NLEV2+I+M)= X(I)/SIGMA
90    Y(NLEV2+I)= Y(I)/SIGMA
      X(NLEVS+1)= 1000.
      IF(M.EQ.1) Y(NLEVS)= TEMP/SIGMA
      DO 100 I=1,NLEV2+M
      X(I)= -X(NLEVS+2-I)
100   Y(I)= -Y(NLEVS+1-I)
C
      RETURN
      END
C
C
^L
```



```
C
C
      SUBROUTINE CENTRD(P,MAX,XBEG,Q,XEND)
C
C  COMPUTES THE CENTROID OF A HISTOGRAM WITHIN A SPECIFIED
C  INTERVAL.  DETAILS OF THE NUMERICAL INTEGRATION METHOD
C  ARE GIVEN IN STROH AND PAEZ, "A COMPARISON OF OPTIMUM AND
C  LOGARITHMIC QUANTIZATION FOR SPEECH PCM AND DPCM SYSTEMS,"
C  IEEE TRANS. ON COMMUNICATIONS, JUNE 1973.
C
C    INPUTS:
C       P       HISTOGRAM.  P(K)= PROBABILITY OR NUMBER OF
C               OBSERVATIONS OF X IN THE KTH "BIN", K-1 < X < K.
C       MAX     NUMBER OF HISTOGRAM BINS.
C    XBEG,XEND  ENDPOINTS OF THE INTERVAL OVER WHICH THE CENTROID
C               IS TO BE COMPUTED.  0 < XBEG < XEND < MAX.
C    OUTPUTS:
C       Q       CENTROID (XBEG < Q < XEND).
C
      DIMENSION P(MAX)
C
      IXBEG= IFIX(XBEG)+1
      IXEND= IFIX(XEND)+1
C
      FN= .5*(XEND*XEND*P(IXEND)-XBEG*XBEG*P(IXBEG))
      FD= XEND*P(IXEND)-XBEG*P(IXBEG)
C
      IF(IXEND.EQ.IXBEG) GO TO 75
C  XBEG AND XEND NOT IN THE SAME BIN.
      FN= FN+ .5*(FLOAT(IXBEG**2)*P(IXBEG)-FLOAT((IXEND-1)**2)
     1*P(IXEND))
      FD= FD+ FLOAT(IXBEG)*P(IXBEG) -FLOAT(IXEND-1)*P(IXEND)
C
      IF(IXEND.LE.IXBEG+1) GO TO 75
C  XBEG AND XEND NOT IN ADJACENT BINS.
      DO 50 II=IXBEG+1,IXEND-1
      FN= FN+ (FLOAT(II)-.5)*P(II)
50    FD= FD+ P(II)
C
75    Q= FN/FD
C
      RETURN
      END
C
C
^L
```

```
C
C
      SUBROUTINE XMEAT(P,MAX,XBEG,Q,XEND,IFLAG)
C
C  COMPUTES THE END POINT OF A HISTOGRAM INTERVAL WHOSE STARTING
C  POINT AND CENTROID ARE SPECIFIED.
C
C    INPUTS:
C       P       HISTOGRAM.  P(A)= PROBABILITY OR NUMBER OF
C               OBSERVATIONS OF X IN THE KTH "BIN", A-1 < X < A.
C       MAX     NUMBER OF HISTOGRAM BINS.
C       XBEG    STARTING POINT OF INTERVAL (0 < XBEG < Q).
C       Q       CENTROID OF INTERVAL (XBEG < Q < MAX).
C    OUTPUTS:
C       XEND    ENDING POINT OF INTERVAL.
C       IFLAG   SET IN CASE THE ENDING POINT OF THE INTERVAL LIES
C               IN A BIN WHOSE INDEX IS GREATER THAN MAX.
C
      DIMENSION P(MAX)
C
C  FIRST, DETERMINE WHICH HISTOGRAM BIN CONTAINS XEND.
C  STARTING WITH THE BIN CONTAINING XBEG, COMPUTE THE
C  CENTROID FROM XBEG TO THE ENDPOINT OF THAT BIN.
      IFLAG=0
      N1= IFIX(XBEG)+1
      II= N1
      TRY= FLOAT(N1)
      FD= (TRY-XBEG)*P(N1)
      FN= .5*(TRY*TRY-XBEG*XBEG)*P(N1)
C
C  CHECK WHETHER THE TEST CENTROID IS GREATER THAN Q.
C  IF SO, JUMP; IF NOT, TRY THE NEXT BIN.
10    QTEST= FN/FD
      IF(QTEST.GT.Q) GO TO 100
      II= II+1
      IF(II.LE.MAX) GO TO 20
      IFLAG=-1
      RETURN
20    FD= FD +P(II)
      FN= FN +(FLOAT(II)-.5)*P(II)
      GO TO 10
100   N2= II
C
C  USING THE APPROXIMATIONS OF STRUM AND PAEZ, THE
C  EQUATION Q=FN(XBEG,XEND)/FD(XBEG,XEND) CAN BE SOLVED
C  FOR XEND.  THE RESULT IS A QUADRATIC EQUATION WITH
C  COEFFICIENTS A, B, AND C.
      A= .5*P(N2)
      B= -Q*P(N2)
      C= -.5*P(N1)*XBEG*XBEG +Q*P(N1)*XBEG
```



```
C
      IF(N2.EQ.N1) GO TO 75
      C= C +.5*(FLOAT(N1*N1)*P(N1) -(FLOAT(N2-1)**2)*P(N2))
     1     -Q*(FLOAT(N1)*P(N1) -FLOAT(N2-1)*P(N2))
C
      IF(N2.LE.N1+1) GO TO 75
      DO 50 II= N1+1,N2-1
50    C= C +(FLOAT(II)-.5-Q)*P(II)
C
75    XEND= (-B +SQRT(B*B-4.*A*C))/(2.*A)
C
      RETURN
      END
C
C
^L
```

```
      SUBROUTINE FULBC(IR,NFRAME,IFULB)
C
C  CONVERT FROM NATURAL BINARY TO FOLDED
C  BINARY CODE, OR VICA-VERSA.  IR IS AN
C  INPUT/OUTPUT VECTOR OF LENGTH NFRAME,
C  WHICH CONTAINS THE CODE WORDS.
C
C
      DIMENSION IR(1)
C
      DO 10 I=1,NFRAME
      IF(IR(I).LE.IFULB) IR(I)=IFULB-IR(I)
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE EXTRNB(NDATT,IDAT,NP,NB,NWW,INW,NWWT,IDATS,IBS
     1,ANW,ER)
C  ROUTINE TO EXTRACT BITS TO BE PROTECTED FROM CODED PARAMETERS
C  FOUR-BIT WORDS ARE FORMED PRIOR TO ERROR PROTECTION
C  VIA HAMMING(7,4) OR (8,4) CODE.
C
C  INPUTS:
C       IDAT    VECTOR OF QUANTIZED LEVELS OF A PRESPECIFIED
C               SEQUENCE OF TRANSMITTED PARAMETERS.
C       NP      VECTOR GIVING THE NUMBER OF BITS PROTECTED
C               CORRESPONDING TO THE SEQUENCE IN IDAT
C       NB      VECTOR GIVING TOTAL NUMBER OF BITS FOR EACH
C               PARAMETER(SEQUENCE AS IN IDAT)
C       ER      VECTOR OF PERCENT CHANNEL BIT ERROR FOR EACH PARAMETER
C               (SEQUENCE AS IN IDAT)
C       NDATT   TOTAL NUMBER OF PARAMETERS
C       NWWT    TOTAL NUMBER OF 4-BIT WORDS TO BE ERROR PROTECTED
C  OUTPUTS:
C       NWW     COUNTER OF THE NUMBER OF 4-BIT WORDS FORMED
C       INW     VECTOR OF 4-BIT WORDS TO BE ERROR PROTECTED
C       ENW     VECTOR OF PERCENTAGE OF CHANNEL BIT ERROR
C               FOR EACH 4-BIT WORD IN INW
C       IDATS   VECTOR OF SUB WORDS(I.E. WHAT IS LEFT
C               AFTER EXTRACTING PROTECTED BITS FROM
C               PARAMETER COMES)
C       IBS     VECTOR OF BITS/SUB WORD
C
C
      DIMENSION IDAT(1),NP(1),NB(1),INW(1),IDATS(1),IBS(1)
      DIMENSION IB(4),ERN(1),ER(1)
C
      COMMON/INASK/INASK(32)
C
      DATA ICONST/4/
C
C  INITIALIZE COUNTS
      NWW=0
      NDAT=1
      INDEX=0
C  ZERO 4 BIT WORDS
      CALL SETZRO(INW(1),NWWT)
      IF(NDATT.EQ.0) RETURN
C  SET UP FOR NEW PARAMETER
C  PUT DATA,#PROTECTED BITS, & TOTAL BITS IN DUMMY VARIABLES
10    IDUM=IDAT(NDAT)
      IPDUM=NP(NDAT)
      IBDUM=NB(NDAT)
      ERDUM=ER(NDAT)
      IF(NP(NDAT).EQ.0) GO TO 30
C  SET INDEX TO 1 - # PROTECTED BITS
      IP=1
```



```
C  INDEX BIT COUNT - 4 BIT WORDS
15    INDEX=INDEX+1
C  EXTRACT BITS FROM DATA - FOR 4 BIT WORD
      IN(INDEX)=0
      IF((IDUM.AND.INASK(IBDUM-IP+1)).EQ.0) GO TO 20
C  BIT EXTRACTED IS 1 - SET THIS BIT IN IDUM=0
      IDUM=IDUM-INASK(IBDUM-IP+1)
C  TURN ON IB
      IB(INDEX)=1
20    CONTINUE
C  CHECK IF 4 BITS COLLECTED
      IF(INDEX.LT.ICONST) GO TO 40
C  4 BITS COLLECTED - MAKE 4 BIT OUTPUT WORD
      J=ICONST
      NWW=NWW+1
      DO 30 I=1,ICONST
      J=J-1
      INW(NWW)=INW(NWW)+LSH(IB(I),J)
30    CONTINUE
      ERN(NWW)=ERDUM
C  RESET 4 BIT WORD INDEX - READY NEW WORD
      INDEX=0
C  CHECK IF PARAMETERS FINISHED
40    IF(IP.EQ.IPDUM) GO TO 50
C  WORD NOT FINISHED CONTINUE
      IP=IP+1
      GO TO 15
C  WORD FINISHED - SET UP SUB WORD & SUB WORD BITS
50    IDATS(NDAT)=IDUM
      IBS(NDAT)=IBDUM-IPDUM
      IF(NDAT.EQ.NDATT) RETURN
C  NOT DONE - GET NEW PARAMETER
      NDAT=NDAT+1
      GO TO 10
      RETURN
      END
^L
```

```
      SUBROUTINE CODN(IWD,NOPT,IWORD)
C
C  CONVERTS 4-BIT WORDS INTO (7,4) OR (8,4)HAMMING CODE WORDS
C
C  INPUTS:
C       IWD     4-BIT DATA WORD TO BE CODED
C       NOPT    SWITCH TO SPECIFY (8,4) OR (7,4) HAMMING CODE WORD
C               3, (7,4) HAMMING CODING
C               4, (8,4) HAMMING CODING
C  OUTPUT:
C       IWORD   (7,4) OR (8,4) HAMMING CODE WORD
C
      DIMENSION IB(4),IC(4),IW(8)
      EQUIVALENCE(IW(1),IB(1)),(IC(1),IW(5))
      COMMON/IHAMWD/IHAMWD(8)
      COMMON/IMASK/IMASK(32)
      DATA ICONST/4/
C
C
C  INITIALIZE
      IBYTE=IWD
      IWORD=0
C  EXTRACT BITS
      DO 10 I=1,ICONST
      IB(I)=0
      IF(((IBYTE.AND.IMASK(ICONST-I+1)).EQ.0) GO TO 10
C  BIT IS 1 - TURN ON IB
      IB(I)=1
10    CONTINUE
C  COMPUTE CHECK BITS
      IC(1)=MOD((IB(1)+IB(2)+IB(4)),2)
      IC(2)=MOD((IW(1)+IW(3)+IW(4)),2)
      IC(3)=MOD((IB(2)+IB(3)+IB(4)),2)
C  FOR (8,4)  CODE COMPUTE AN ADDITIONAL PARITY CHECK BIT
      IF(NOPT.EQ.4)IC(4)=MOD((IB(1)+IC(1)+IC(2)),2)
C  MAKE HAM CODE WORD FROM IW=>IB+IC
      DO 20 I=1,NOPT+4
      IF((IW(I).EQ.1) IWORD=IWORD.OR.IHAMWD(I)
20    CONTINUE
      RETURN
      END
^L
```



```
      SUBROUTINE ERRCHN(IWD,NB,PCTE,IWDOUT,KRESFG)
C
C  CHANGES BITS (0 TO 1, OR 1 TO 0) OF A
C  WORD TO SIMULATE THE EFFECT OF CHANNEL BIT ERRORS
C
C  INPUTS:
C       IWD     INPUT WORD TO BE ERRORED
C       NB      TOTAL NUMBER OF BITS OF IWD
C               WHICH REPRESENT TRANSMITTED DATA.
C       PCTE    PERCENT ERROR FOR EACH IWD
C               (AS DECIMAL NUMBER,.01 FOR ONE PERCENT)
C       KRESFG  FLAG TO INITIATE COUNT OF RESIDUAL BIT ERRORS
C  OUTPUT:
C       IWDOUT  OUTPUT ERRORED WORD
C
C
      COMMON/CUM/ICUMBT,IERCUM,NSERR
C
C  ICUMBT= COUNTER OF TOTAL BITS PROCESSED
C  IERCUM= COUNTER OF TOTAL BITS WHICH HAVE BEEN ERRORED
C  NSERR= COUNTER OF TOTAL NUMBER OF RESIDUAL BITS ERRORED
      COMMON/IMASK/IMASK(32)
      IWDOUT=IWD
      IF(NB.EQ.0) RETURN
      IF(PCTE.EQ.0.) GO TO 50

      DO 10 I=1,NB
C  CALL RANDOM NUMBER GENERATOR FUNCTION
C  (NOTE : THIS FUNCTION NOT PART OF SUPPLIED SOFTWARE
C   USER MUST SUBSTITUTE APPROPIATE ROUTINE)
      X=RANDOM(0.0,1.0)
      IF(X.LT.PCTE) GO TO 20
      GO TO 30
20    IWDOUT=IWDOUT.XOR.IMASK(I)
      IF(KRESFG)NSERR=NSERR+1
      IERCUM=IERCUM+1
30    ICUMBT=ICUMBT+1
10    CONTINUE
      RETURN
C  NO ERROR
50    ICUMBT=ICUMBT+NB
      RETURN
      END
^L
```

```
C  <DCA16>1644R.F4.2   Thu 21-Aug-80 9:39AM          Page 1

      SUBROUTINE DCUDK(IWD,NUPT,IWORD,N84FG)
C  DECODES 7,4 OR 8,4 HAMMING CODE WORD
C  OUTPUT IS A 4-BIT ERROR  CORRECTED WORD
C
C  INPUTS:
C       IWD     7,4 OR 8,4 HAMMING CODE WORD
C       NUPT    SWITCH TO SPECIFY (8,4) OR (7,4) HAMMING CODE WORD
C               3, (7,4) CODE WORD
C               4, (8,4) CODE WORD
C  OUTPUTS:
C       IWORD   4-BIT ERROR CORRECTED WORD
C       N84FG   FLAG TO INDICATE EVEN ERRORS(8,4 CODE ONLY)
C               -1, EVEN ERRORS DETECTED
C               0 , USD ERRORS DETECTED
C
C
C
C
      DIMENSION IB(4),IC(4),IW(8)
      EQUIVALENCE (IB(1),IW(1)),(IC(1),IW(5))
      COMMON/IHAMWD/IHAMWD(8)
      COMMON/IMASK/IMASK(32)
      DATA ICONST/4/
C
C  INITIALIZE
      IDWN=IWD
      IWORD=0
      N84FG=0
C  EXTRACT BITS FROM HAM WORD
      DO 10 I=1,NUPT+4
      IW(I)=0
      IF((IDWN.AND.IHAMWD(I)).NE.0) IW(I)=1
10    CONTINUE
C  IF 7,4 CODE - GO COMPUTE CHECK BITS
      IF(NUPT.NE.4) GO TO 40
C  8,4 CODE - GET PARITY BIT
      ISUM=0
      DO 20 I=1,8
      ISUM=ISUM+IW(I)
20    CONTINUE
      IC(4)=MOD(ISUM,2)
C  COMPUTE CHECK BITS
40    IC(1)=MOD((IW(1)+IC(1)+IW(2)+IW(4)),2)
      IC(2)=MOD((IB(1)+IC(2)+IB(3)+IB(4)),2)
      IC(3)=MOD((IB(2)+IC(3)+IB(3)+IB(4)),2)
C  COMPUTE CORRECTION BIT LOCATION
      ICOR=LSH(IC(3),2)+LSH(IC(2),1)+IC(1)
C  FLIP BIT IF ERROR - 7,4 OR 8,4
      IF(ICOR.GT.0) IDWN=IDWN.XOR.IMASK(ICOR)
C  IF 8,4 TURN ON FLAG IF TWO ERRORS
      IF((NUPT.EQ.4).AND.(IC(4).EQ.0).AND.(IC(R.NE.0)) N84FG=-1

C  <DCA16>16KER.F4.2   Thu 21-Aug-80 9:39AM          Page 1:1

C  EXTRACT 4 DATA BITS FROM HAM WORD
      J=ICONST
      DO 50 I=1,ICONST
C     TURN ON IWORD BIT # J (IE SET BIT J TO 1)
      IWORD=IWORD.OR.IMASK(J)
C     TURN OFF IWORD BIT J IF IDWN I BIT IS ZERO
      IF((IDWN.AND.IHAMWD(I)).EQ.0)IWORD=IWORD.XOR.IMASK(J)
      J=J-1
50    CONTINUE
      RETURN
      END
^L
```

```
      SUBROUTINE COMBWD(IWw,NP,NB,NDATT,IDATS,IDAT)
C
C  COMBINES 4-BIT ERROR CORRECTED WORDS
C  AND SUB WORDS TO RECREATE PARAMETER CODES
C
C  INPUTS:
C       IWW     VECTOR OF ERROR CORRECTED 4-BIT WORDS
C       NP      VECTOR OF NUMBER OF PROTECTED BITS OF PARAMETER CODES
C       NB      VECTOR OF NUMBER OF BITS OF PARAMETER CODES
C       NDATT   NUMBER OF PARAMETER CODES
C       IDATS   VECTOR OF SUB WORDS
C               (NOTE : ALL ARRAYS ARE OF A PRESPECIFIED SEQUENCE)
C  OUTPUTS:
C       IDAT = VECTOR REGENERATED PARAMETER CODES
C
C
      DIMENSION INd(1),NP(1),Nd(1),IDATS(1),IDAT(1)
      DIMENSION IB(4)
      COMMON/IMASK/IMASK(32)
      DATA ICONST/4/
C
C  SET INDEXES
      L=0
      K=0
      J=NB(1)
      NWW=1
      NBd=1
      ICNST1=ICONST+1
C  ZERO DATA WORDS
      CALL SETZM(IDAT(1),NDATT)
C SEPERATE 4-BIT WORDS INTO SINGLE BIT WORDS IB
10    CONTINUE
      DO 20 I=1,4
      IB(I)=0
      IF((INd(NWW).AND.IMASK(ICNST1-I)).EQ.0) GO TO 20
      IB(I)=1
20    CONTINUE
      IF(NP(NDw).EQ.0) GO TO 50
C  LOOP ON FORMATION OF DATA WORDS
40    J=J-1
      K=K+1
      L=L+1
      IDAT(NWw)=IDAT(NBd)+LSw(IB(K),J)
C  CHECK IF PROTECTED BITS COMPLETE
      IF(L.LT.NP(NDW)) GO TO 30
C  PROTECTED BITS COMPLETE - COMPLETE WORD
      GO TO 50
C  CHECK IF ANY SINGLE BIT WORDS  LEFT
30    IF(K.LT.ICONST) GO TO 40
C  OUT OF SINGLE BIT WORDS - GET MORE IB'S
      NBd=NBd+1
      K=0
      GO TO 10
```



```
C  COMPLETE WORD
50    IDAT(NWw)=IDAT(NWw)+IDATS(NDw)
C  IF COMPLETE RETURN
      IF(NDW.EQ.NDATT) RETURN
C  MORE LEFT TO DO
60    NDw=NDw+1
      J=NB(NDW)
      L=0
      IF(NP(NDw).EQ.0) GO TO 50
C  CHECK IF SINGLE BIT WORDS USED UP
      GO TO 30
      RETURN
      END
^L
```

```
      SUBROUTINE SEQTX(NDATT,ISEQ,ICW,IPTCH,IENQ,IENS,IPTAP,IR,IDAT
     1,ER,ERP)
C
C  PUTS PARAMETER CODES AND CHANNEL ERROR PERCENT
C  FOR EACH PARAMETER INTO THE ARRAYS IDAT AND ER
C  RESPECTIVELY. SEQUENCE OF CODE AND ERROR STORAGE IN THESE ARRAYS
C  IS PREDEFINED BY THE ARRAY ISEQ. ISEQ CONTAINS
C  NUMBERS WHICH CORRESPOND TO THE PARAMETER TYPES AS GIVEN BELOW.
C
C  INPUTS:
C       NDATT    TOTAL NUMBER OF PARAMETER CODES
C       ISEQ     VECTOR OF NUMBERS DESCRIBING THE SEQUENCE OF
C                PARAMETER CODES
C       ERP      PERCENT OF CHANNEL ERROR FOR EACH PARAMETER TYPE
C                ERP(1) - REFLECTION COEFFS
C                ERP(2) - PITCH
C                ERP(3) - OVERALL GAIN
C                ERP(4) - DELTA GAINS
C                ERP(5) - PITCH PREDICTOR TAPS
C                ERP(6) - RESIDUAL SAMPLES
C       SINGLE PARAMETER CODES
C                TYPE             VARIABLE         ISEQ #
C                ----             --------         ------
C                REFLN COEFF      ICW              1 TO 14
C                PITCH            IPTCH            15
C                GAIN             IENQ             16 TO 10
C                DELTA GAINS      IENS             21 TO 30
C                PITCH TAPS       IPTAP            31 TO 35
C                RES SAMPLS       IR               36 ---
C
C  OUTPUTS:
C       IDAT     VECTOR OF PARAMETER CODES
C                SEQUENCE AS IN ISEQ
C       ER       VECTOR OF PERCENT ERROR FOR EACH PARAMETER CODE
C                SEQUENCE AS IN ISEQ
C
C
      DIMENSION ISEQ(1),ICW(1),IENS(1),IPTAP(1),IR(1)
      DIMENSION ER(1),ERP(1),IDAT(1)
C
C
      DO 1000 I=1,NDATT
      IPRAM=ISEQ(I)
C  REFLECTION COEFFS
      IF(IPRAM.LE.14) GO TO 10
C  PITCH
      IF(IPRAM.EQ.15) GO TO 20
C  GAIN
      IF(IPRAM-15.LE.5) GO TO 30
C  DELTA GAINS
      IF(IPRAM-20.LE.10) GO TO 40
```



```
C  PITCH TAPS
      IF(IPRAM-30.LE.5) GO TO 50
      GO TO 60
C
C
C  REFLECTION COEFFS
10    IDAT(I)=ICW(IPRAM)
      ER(I)=ERP(1)
      GO TO 1000
C  PITCH
20    IDAT(I)=IPTCH
      ER(I)=ERP(2)
      GO TO 1000
C  GAIN
30    INDX=IPRAM-15
      IDAT(I)=IENQ
      ER(I)=ERP(3)
      GO TO 1000
C  DELTA GAINS
40    INDX=IPRAM-20
      IDAT(I)=IENS(INDX)
      ER(I)=ERP(4)
      GO TO 1000
C  PITCH TAPS
50    INDX=IPRAM-30
      IDAT(I)=IPTAP(INDX)
      ER(I)=ERP(5)
      GO TO 1000
C  RESIDUAL SAMPLES
60    INDX=IPRAM-35
      IDAT(I)=IR(INDX)
      ER(I)=ERP(6)
1000  CONTINUE
      RETURN
      END
^L
```

```
      SUBROUTINE SEQNX(NDATT,ISEQ,ICQ,IPTCH,IENQ,IENS,IPTAP,IR,IDAT)
C
C  DERIVE INDIVIDUAL PARAMETERS OR RESIDUAL SAMPLES
C  FROM THE COMPOSITE ARRAY IDAT.(SEE SEQTX FOR DEFINATION
C  OF ARGUMENTS)
C
      DIMENSION ISEQ(1),ICQ(1),IENS(1),IPTAP(1),IR(1)
      DIMENSION IDAT(1)
C
C
      DO 1000 I=1,NDATT
      IPRAM=ISEQ(I)
C  REFLECTION COEFFS
      IF(IPRAM.LE.14) GO TO 10
C  PITCH
      IF(IPRAM.EQ.15) GO TO 20
C  GAIN
      IF(IPRAM-15.LE.5) GO TO 30
C  DELTA GAINS
      IF(IPRAM-20.LE.10) GO TO 40
C  PITCH TAPS
      IF(IPRAM-30.LE.5) GO TO 50
      GO TO 60
C
C
C  REFLECTION COEFFS
10    ICQ(IPRAM)=IDAT(I)
      GO TO 1000
C  PITCH
20    IPTCH=IDAT(I)
      GO TO 1000
C  GAIN
30    IENQ=IDAT(I)
      GO TO 1000
C  DELTA GAINS
40    INDX=IPRAM-20
      IENS(INDX)=IDAT(I)
      GO TO 1000
C  PITCH TAPS
50    INDX=IPRAM-30
      IPTAP(INDX)=IDAT(I)
      GO TO 1000
C  RESIDUAL SAMPLES
60    INDX=IPRAM-35
      IR(INDX)=IDAT(I)
1000  CONTINUE
      RETURN
      END
^L
```



```
      SUBROUTINE BSTRMI(IR,NBPS,NSMP,NST,IB,NBITT)
C
C  BIT STREAMING OF THE RESIDUAL SAMPLES
C
C  EXTRACTS BITS FROM THE RESIDUAL SAMPLE PARAMETER CODES
C  (IN ARRAY IR) AND FORMS AN ARRAY IB OF RESIDUAL
C  SAMPLE BITS (1 DATA BIT /WORD)
C
C  INPUTS:
C       IR      VECTOR OF RESIDUAL SAMPLE CODES
C       NBPS    VECTOR OF BITS PER RESIDUAL SAMPLE
C               (THIS ARRAY ALLOWS A VARIABLE # OF BITS/SAMPLE)
C       NSMP    TOTAL NUMBER OF RESIDUAL SAMPLE CODES
C       NST     INDEX OF THE FIRST ENTRY IN NBPS
C  OUTPUTS:
C       IB      VECTOR OF RESIDUAL SAMPLE BITS(1 DATA BIT / WORD)
C       NBITT   TOTAL NUMBER OF 1 DATA BIT WORDS IN IB
C
C
C
      DIMENSION IR(1),NBPS(1),IB(1)
      COMMON/IMASK/IMASK(32)
C
C
C  COUNT ON BITS TOTAL
      NBITT=0
C  INDEX FOR BITS/SAMPLE
      II=NST-1
C  LOOP FOR FRAME
      DO 100 I=1,NSMP
C  UPDATE INDEX FOR BITS/SAMPLE
      II=II+1
C  CHECK IF ZERO BITS USED - IF SO GO TO NEXT SAMPLE
      IF(NBPS(II).EQ.0) GO TO 100
C  BITS NOT ZERO - PUT SAMPLE IN DUMMY VARIABLE
      IDDUM=IR(I)
C  INITIALIZE BITS/SAMPLE COUNTER
      IBIT=1
C  UPDATE TOTAL BIT COUNTER
15    NBITT=NBITT+1
C  INITIALIZE BIT ARRAY
      IB(NBITT)=0
      IF((IDDUM.AND.IMASK(NBPS(II)-IBIT+1)).EQ.0) GO TO 20
C  BIT EXTRACTED IS 1 - SET THIS BIT TO ZERO IN IDDUM
      IDDUM=IDDUM-IMASK(NBPS(II)-IBIT+1)
C  TURN ON IB
      IB(NBITT)=1
20    CONTINUE
      IF(IBIT.EQ.NBPS(II)) GO TO 100
      IBIT=IBIT+1
      GO TO 15
100   CONTINUE
      RETURN
      END
^L
```

```
      >SUBROUTINE BSTRMK(IBR,NBITT,NBPS,NST,IRR,NSBBC,NSMP)
C
C  FORMS RESIDUAL SAMPLE CODES FROM
C  THE BIT STREAM FORMED IN BSTRMI
C
C  INPUTS:
C       IBR     VECTOR OF RESIDUAL SAMPLE BITS(1 DATA BIT / WORD)
C       NBITT   TOTAL NUMBER OF 1 BIT DATA WORDS IN IBR
C       NBPS    VECTOR OF # OF BITS /RESIDUAL SAMPLE CODE
C       NST     INDEX OF FIRST ENTRY IN NBPS
C       NSBBC   TOTAL NUMBER OF RESIDUAL SAMPLE CODES
C  OUTPUTS:
C       NSMP    COUNTER OF NUMBER OF RESIDUAL SAMPLE CODES
C       IRR     VECTOR OF RESIDUAL SAMPLE CODES
C
C
      DIMENSION IBR(1),NBPS(1),IRR(1)
      COMMON/LMASK/LMASK(32)
C
C
C  INDEX ON BITS/SAMPLE ARRAY
      II=NST
C  INITIALIZE SAMPLE COUNT
      NSMP=1
C  INDEX ON BITS/SAMPLE
10    J=NBPS(II)-1
C  SET SAMPLE TO ZERO
      IRR(NSMP)=0
C  CHECK IF BITS = 0 - IF SO DONT ENTER BIT STREAM LOOP
C  LOOK AT NEXT SAMPLE
      IF(J.GE.0) GO TO 20
      II=II+1
      NSMP=NSMP+1
      GO TO 10
C  BIT STREAM LOOP
20    CONTINUE
      DO 100 I=1,NBITT
      IRR(NSMP)=IRR(NSMP)+LSH(IBR(I),J)
      J=J-1
      IF(J.EQ.-1) GO TO 50
      GO TO 100
C  SAMPLE FINISHED - RESET
50    NSMP=NSMP+1
      IRR(NSMP)=0
      II=II+1
      J=NBPS(II)-1
C  IF TOT NUM OF SAMPS - JUMP OUT
      IF(NSMP.GT.NSBBC) GO TO 110
C  CHECK IF NEXT SAMP HAS ZERO BITS
```



```
C  IF SO DONT INDEX BIT LOOP - GO TO NEXT SAMPLE
      IF(J.GE.0) GO TO 100
      GO TO 50
100   CONTINUE
110   NSMP=NSMP-1
      RETURN
      END
^L
```

```
      FUNCTION ENERGY(S,MAXS,NST,NEN)
C
C  COMPUTES ENERGY OR MEAN-SQUARED VALUE.
C
C    ARGUMENTS:
C       S       VECTOR OF SAMPLES.
C       MAXS    MAXIMUM DIMENSION OF S.
C       NST,NEN INDICES OF THE FIRST AND LAST SAMPLES FOR WHICH
C               THE MEAN-SQUARED VALUE IS TO BE COMPUTED.
C
C
      DIMENSION S(MAXS)
C
      ENERGY=0.0
      NTOT=NEN-NST+1
      DO 10 I=NST,NEN
10    ENERGY=ENERGY+S(I)*S(I)
      ENERGY=ENERGY/FLOAT(NTOT)
      RETURN
      END
^L
```



```
C
      SUBROUTINE ZERODC(S,MAXS,NST,NEN,TEMP)
C
C  COMPUTES AND SUBTRACTS DC OR MEAN VALUE.
C
C    INPUTS:
C       S       VECTOR OF INPUT SAMPLES
C               (DC-REMOVED OUTPUT STORED ALSO IN S).
C       MAXS    DIMENSION OF S.
C       NST,NEN INDICES OF THE FIRST AND LAST SAMPLES FOR WHICH
C               THE MEAN IS TO BE REMOVED.
C    OUTPUT:
C       TEMP    MEAN VALUE.
C
C
      DIMENSION S(MAXS)
C
      TEMP=0.0
      NTOT=NEN-NST+1
      DO 10 I=NST,NEN
10    TEMP=TEMP+S(I)
      TEMP=TEMP/FLOAT(NTOT)
      DO 20 I=NST,NEN
20    S(I)=S(I)-TEMP
      RETURN
      END
^L
```

```
C
      SUBROUTINE PREEM(X,Y,MAX,NST,NEN,PEMFAC)
C
C  1ST ORDER PREEMPHASIS.
C
C    INPUTS:
C       X       VECTOR OF INPUT SAMPLES
C       MAX     MAXIMUM DIMENSION OF X AND Y
C       NST,NEN INDICES OF THE FIRST AND LAST SAMPLES FOR WHICH
C               PREEMPHASIS IS TO BE PERFORMED.
C       PEMFAC  PREEMPHASIS FILTER CONSTANT(1-PEMFAC(Z**-1))
C    OUTPUT:
C       Y       VECTOR OF PREEMPHASIZED SPEECH SAMPLES
C
C
C
      DIMENSION X(MAX),Y(MAX)
      DO 10 I=NEN,NST,-1
10    Y(I)= X(I)-PEMFAC*X(I-1)
      RETURN
      END
^L
```



```
C
      SUBROUTINE DEMPH(S,MAXS,NST,NEN,DEMFAC)
C
C  1ST ORDER DEEMPHASIS.
C
C    INPUTS:
C       S       VECTOR OF INPUT SAMPLES.
C               (DEEMPHASIZED SAMPLES ARE ALSO STORED IN S)
C       MAXS    MAXIMUM DIMENSION OF S
C       NST,NEN INDICES OF THE FIRST AND LAST SAMPLES FOR WHICH
C               DEEMPHASIS IS TO BE PERFORMED.
C       PEMFAC  DEEMPHASIS FILTER CONSTANT.
C
C
      DIMENSION S(MAXS)
C
      DO 10 I=NST,NEN
10    S(I)=S(I)+DEMFAC*S(I-1)
      RETURN
      END
^L
```

```
C
      SUBROUTINE RES(S,MAXS,R,MAXR,A,NSTR,NP,NST,NEN)
C
C
C  LPC (SPECTRAL) INVERSE FILTERING TO COMPUTE RESIDUAL
C
C    INPUTS:
C       S       VECTOR OF INPUT (SPEECH) SAMPLES.
C       MAXS    DIMENSION OF S.
C       MAXR    DIMENSION OF R.
C       A       PREDICTOR COEFFICIENTS.
C       NSTR    INDEX OF FIRST OUTPUT SAMPLE.
C       NP      NUMBER OF POLES (=LENGTH OF A).
C       NST,NEN INDICES OF THE FIRST AND LAST INPUT SAMPLES
C               TO BE FILTERED.
C    OUTPUT:
C       R       VECTOR OF RESIDUAL SAMPLES.
C
C
      DIMENSION S(MAXS),R(MAXR),A(NP)
C
      M=NSTR-NST
      DO 100 I=NST,NEN
      SHAT=0.0
      DO 10 K=1,NP
      J=I-K
10    SHAT=SHAT-A(K)*S(J)
      KK=M+I
100   R(KK)=S(I)-SHAT
      RETURN
      END
^L
```



```
C
      SUBROUTINE SYNTH(S,MAXS,A,R,MAXR,NSTR,NP,NST,NEN)
C
C
C  LPC (SPECTRAL) SYNTHESIS.
C
C    INPUTS:
C       R       VECTOR OF EXCITATION SAMPLES.
C       MAXR    MAXIMUM DIMENSION OF R.
C       MAXS    MAXIMUM DIMENSION OF S.
C       A       PREDICTOR COEFFICIENTS.
C       NSTR    INDEX OF FIRST INPUT SAMPLE.
C       NP      NUMBER OF POLES (=LENGTH OF A).
C       NST,NEN INDICES OF THE FIRST AND LAST OUTPUT SAMPLES
C    OUTPUT:
C       S       VECTOR OF SYNTHESIZED (SPEECH) SAMPLES.
C
C
      DIMENSION S(MAXS),A(NP),R(MAXR)
C
      M=NSTR-NST
      DO 100 I=NST,NEN
      J=M+I
      TEMP=-R(J)
      JJ=0
      DO 10 KK=1,NP
      JJ=JJ+1
      IMJJ=I-JJ
10    TEMP=TEMP+A(KK)*S(IMJJ)
      S(I)=-TEMP
100   CONTINUE
      RETURN
      END
^L
```

```
      SUBROUTINE HAMWND(S,MAXS,NST,NEN,SWN)
C
C
C  HAMMING WINDOW.
C
C    INPUTS:
C       S       VECTOR OF INPUT SAMPLES.
C       MAXS    DIMENSION OF S.
C       NST,NEN INDICES OF THE FIRST AND LAST INPUT SAMPLES
C               TO BE WINDOWED.
C    OUTPUT:
C       SWN     VECTOR OF HAMMING WINDOWED SAMPLES.
C
C
      DIMENSION S(MAXS),SWN(MAXS)

      COMMON/ALPHA/ALPHA,ALPHA1
      COMMON/PI/PI,TWOPI
C
      NPTS=NEN-NST+1
      PI2N=TWOPI/FLOAT(NPTS)
      FI=-1.0
      DO 10 I=NST,NEN
      FI=FI+1.0
      SWN(I)=S(I)*(ALPHA-ALPHA1*COS(FI*PI2N))
10    CONTINUE
      RETURN
      END
^L
```



```
      SUBROUTINE AUTCOR(S,MAXS,AR,MAXAR,NST,NEN,NP)
C
C  COMPUTE AUTOCORRELATION FUNCTION.
C
C    INPUTS:
C       S       VECTOR OF (WINDOWED) SAMPLES.
C       MAXS    DIMENSION OF S.
C       MAXAR   DIMENSION OF AR.
C       NST,NEN INDICES OF THE FIRST AND LAST SAMPLES FOR WHICH
C               THE AUTOCORRELATION FUNCTION IS COMPUTED.
C       NP      MAXIMUM LAG OF INTEREST.
C    OUTPUTS:
C       AR      AUTOCORRELATION COEFFICIENTS.
C               NOTE THAT AR(1) IS THE ZEROTH AUTOCORRELATION
C               COEFFICIENT AND AR(NP+1) IS THE AUTOCORRELATION
C               COEFFICIENT FOR THE LAG NP.
C
      DIMENSION S(MAXS),AR(MAXAR)
C
      NPP1=NP+1
      DO 100 J=1,NPP1
      MLOOP=NEN-J+1
      SUM=0.0
      DO 50 N=NST,MLOOP
      N2=N+J-1
      SUM=SUM+S(N)*S(N2)
50    CONTINUE
      AR(J)=SUM
100   CONTINUE
      RETURN
      END
^L
```

```
C
      SUBROUTINE ALPCON(M,IFLAG,R,C,A,V,SCR)
C
C  GIVEN ONE PARAMETER REPRESENTATION
C  OF THE LPC FILTER, IT PROVIDES TWO OTHER
C  REPRESENTATIONS ALONG WITH THE NORMALIZED ERROR.
C  ARGUMENTS:
C  M     =LPC ORDER (NO. OF POLES)
C  IFLAG=1, GIVEN R, THE QUANTITIES C,A AND V ARE COMPUTED;
C       =2, GIVEN C, THE QUANTITIES R,A AND V ARE COMPUTED;
C       =3, GIVEN A, THE QUANTITIES R,C AND V ARE COMPUTED;
C       =-3, GIVEN A, THE QUANTITIES C AND V ARE COMPUTED.
C  R     =AUTOCORRELATION VECTOR OF R(1),...,R(M),R(M+1)
C        (NOTE THE SHIFT IN THE SUBSCRIPT BY ONE.)
C  C     =REFLECTION COEFF. VECTOR OF C(1),C(2),...,C(M)
C  A     =PREDICTION COEFF. VECTOR OF A(1),A(2),...,A(M)
C  V     =NORMALIZED PREDICTION ERROR
C  SCR   =SCRATCH VECTOR OF SCR(1),SCR(2),...,SCR(M)
C
      DIMENSION R(M),C(M),A(M),SCR(M)
C
      GO TO (10,40,70), IFLAG
      GO TO 70
C
C  GIVEN M, COMPUTE C,A AND V
10    A(1)=-R(2)/R(1)
      C(1)=A(1)
      V=R(1)+A(1)*R(2)
      IF(M.EQ.1) GO TO 30
      DO 25 I=2,M
      S=R(I+1)
      IM1=I-1
      DO 15 K=1,IM1
      IMK=I-K
      SCR(K)=A(IMK)
15    S=S+SCR(K)*R(K+1)
      X=-S/V
      DO 20 K=1,IM1
20    A(K)=A(K)+X*SCR(K)
      A(I)=X
      C(I)=X
25    V=V+X*S
30    V=V/R(1)
      RETURN
C
C  GIVEN C, COMPUTE R,A AND V
```



```
40    R(1)=1.0
      A(1)=C(1)
      R(2)=-C(1)
      V=1.0-C(1)**2
      IF(M.EQ.1) GO TO 60
      DO 55 I=2,M
      X=C(I)
      S=-X*V
      IM1=I-1
      DO 45 K=1,IM1
      IMK=I-K
      SCR(K)=A(IMK)
45    S=S-R(K+1)*SCR(K)
      R(I+1)=S
      DO 50 K=1,IM1
50    A(K)=A(K)+X*SCR(K)
      A(I)=X
55    V=V*(1.0-X**2)
60    RETURN
C
C  GIVEN A, COMPUTE C,R AND V
C  SINCE WE WANT TO AVOID THE USE OF A TWO-DIMENSIONAL SCRATCH VECTOR
C  (OR AN EQUIVALENT LONG ONE-DIMENSIONAL VECTOR), WE ACCOMPLISH THE
C  PARAMETER CONVERSION IN TWO STEPS: FIRST, COMPUTE C FROM A; SECOND,
C  GIVEN THIS C, COMPUTE R. THE SECOND STEP IS PERFORMED BY CARRYING OUT
C  THE CASE IFLAG=2 (WHICH GIVES THE VECTOR A ALSO).  FOR IFLAG=-3,
C  WE SIMPLY SKIP THE SECOND STEP.
C
70    C(M)=A(M)
      V=1.0-C(M)**2
      IF(M.EQ.1) GO TO 100
      MM1=M-1
      DO 75 I=1,MM1
      MMI=M-I
75    SCR(I)=(A(I)-A(M)*A(MMI))/V
      IF(M.EQ.2) GO TO 95
      DO 90 K=MM1,2,-1
      KM1=K-1
      DO 80 M=1,K
80    C(M)=SCR(M)
      X=1.-SCR(K)**2
      DO 85 I=1,KM1
      KMI=K-I
85    SCR(I)=(C(I)-SCR(K)*C(KMI))/X
90    V=V*X
95    C(1)=SCR(1)
      V=V*(1.0-C(1)**2)
100   IF(IFLAG.GT.0) GO TO 40
C
      RETURN
      END
^L
^L
```

```
      SUBROUTINE PRDCT(X,XHAT,INDX,F,LF,MPITCH)
C
C  LINEAR PREDICTION OF A SAMPLE VALUE FROM PREVIOUS SAMPLES OF
C  THE SAME SIGNAL.
C
C    INPUTS:
C      X       VECTOR OF INPUT SAMPLES.
C      INDX    INDEX OF THE SAMPLE WHOSE VALUE IS TO BE PREDICTED.
C      F       COEFFICIENTS OF PREDICTION FILTER.
C      LF      NUMBER OF PREDICTION FILTER COEFFICIENTS.
C      MPITCH  IF MPITCH>0, THEN PITCH PREDICTION IS PERFORMED,
C              USING MPITCH AS THE PITCH PERIOD.  IN THIS CASE,
C              F CONTAINS THE PITCH FILTER TAPS AND LF
C              SHOULD BE ODD.  IF MPITCH=0, THEN SPECTRAL
C              LINEAR PREDICTION IS PERFORMED.
C    OUTPUTS:
C      XHAT    PREDICTION OF X(INDX).
C
      DIMENSION X(1),F(1)
C
      IF(MPITCH.EQ.0) M=0
      IF(MPITCH.GT.0) M=MPITCH-(LF+1)/2
      XHAT=0.
      IF(LF.LT.1) GO TO 10
      DO 10 K=1,LF
      I= INDX-M-K
      XHAT= XHAT +F(K)*X(I)
10    CONTINUE
      RETURN
      END
```



```
      SUBROUTINE PITCH(S,NPTS,LOG2N,IFSM,IFSL,MPIT,FR,FI,PN)
C
C  COMPUTE PITCH, USING THE AUTOCORRELATION METHOD.
C
C    INPUTS:
C      S       VECTOR OF INPUT SAMPLES FROM WHICH THE PITCH
C              IS ESTIMATED.
C      NPTS    NUMBER OF SAMPLES OF S.
C      LOG2N   A (2**LOG2N)-POINT FFT IS USED TO EVALUATE THE
C              AUTOCORRELATION FUNCTION.
C      IFSM,   SMALLEST AND LARGEST LAGS CORRESPONDING TO PERMISSIBLE
C      IFSL    VALUES OF PITCH.
C    OUTPUTS:
C      MPIT    PITCH, MEASURED IN NUMBER OF SAMPLES.
C      FR      AUTOCORRELATION COEFFICIENTS (TO BE USED LATER FOR
C              COMPUTING THE PITCH FILTER TAPS).
C    SCRATCH VECTORS:
C      FR,FI   FR, FI, AND PN MUST BE DIMENSIONED AT LEAST
C              2**LOG2N IN THE CALLING PROGRAM.
C
      DIMENSION S(1),FR(1),FI(1),PN(1)
C
      NFFT=2**LOG2N
      NFFT2=NFFT/2
      DO 50 I=1,NPTS
50    FR(I)=S(I)
C  REMOVE MEAN VALUE.
      CALL ZERODC(FR,NPTS,1,NPTS,RMEAN)
C  HAMMING WINDOW.
      CALL HAMWND(FR,NPTS,1,NPTS,FR)
100   DO 1777 I=NPTS+1,NFFT
1777  PN(I)=0.
      DO 7777 I=1,NPTS
7777  PN(I)=FR(I)
C  COMPUTE DFT OF REAL SIGNAL.
C  (FFTR IS PART OF A USM SOFTWARE LIBRARY, AND IS
C  NOT INCLUDED IN THIS SUBROUTINE PACKAGE.)
      CALL FFTR(LOG2N,NFFT,PN,FR,FI)
C  COMPUTE SPECTRUM.
      DO 7799 I=1,NFFT2+1
7799  FR(I)=FR(I)**2+FI(I)**2
C  MAKE SYMMETRIC W.R.T. FR(NFFT2+1), SO THAT
C  INVERSE DFT WILL BE REAL.
      DO 2777 I=2,NFFT2
      N=NFFT+2-I
2777  FR(N)=FR(I)
C  ZERO OUT IMAGINARY PART.
      DO 3777 I=1,NFFT
3777  FI(I)=0.
C  INVERSE DFT.
      CALL FFTR(LOG2N,NFFT,FR,FR,FI)
      MPIT=1+IFSM
      FMAX=0.
```

```
C  SEARCH FOR LARGEST AUTOCORRELATION COEFFICIENT.
      DO 4777 I=IFBN+1,IFBL+1
      IF(FR(I).LT.FMAX) GO TO 4777
      FMAX=FR(I)
      MPIT=I-1
4777  CONTINUE
      RETURN
      END
^Z
```



```
      SUBROUTINE PTAPS(AUTOC,NTAPS,TAPS,MPIT,B0)
C
C  COMPUTE PITCH FILTER TAPS.
C
C    INPUTS:
C      AUTOC   AUTOCORRELATION COEFFICIENTS, WITH
C              AUTOC(1) BEING THE ZEROTH COEFF.
C      NTAPS   NUMBER OF PITCH FILTER TAPS (NO MORE THAN 5).
C      MPIT    PITCH, MEASURED IN NUMBER OF SAMPLES.
C    OUTPUTS:
C      TAPS    FILTER TAPS, SMALLEST LAG FIRST.
C      B0      THE TAP FOR A 1-TAP PITCH FILTER.
C              (TO BE USED IN CASE THE MULTI-TAP CASE
C               IS UNSTABLE)
C
      DIMENSION AUTOC(1),TAPS(1)
      DIMENSION RHS(5),PEO(5)
C
      B0=0.
      CALL SETZM(TAPS,NTAPS)
      IF(NTAPS.LE.0) RETURN
C  EXPLICIT SOLUTION FOR THE 1-TAP CASE
      TAPS(1)=-AUTOC(MPIT+1)/AUTOC(1)
      B0= TAPS(1)
      IF(NTAPS.LE.1) RETURN
      II=1
C  GENERATE THE RIGHT-HAND-SIDE VECTOR RHS
C  OF THE NORMAL EQUATIONS.
      DO 10 I=MPIT-NTAPS/2,MPIT+NTAPS/2
      RHS(II)=-AUTOC(I+1)
      II= II+1
10    CONTINUE
C  LEVINSON RECURSION FOR THE MULTITAP CASE.
      CALL LEVREC(NTAPS,AUTOC,RHS,TAPS,PEO)
      RETURN
      END
^Z
```

```
      SUBROUTINE LEVREC(LR,R,G,F,A)
C  SOLVES AUTOCORRELATION NORMAL EQUATIONS USING LEVINSON RECURSION.
C  PROGRAM FROM RUBINSON'S BOOK, "MULTICHANNEL TIME SERIES ANALYSIS
C  WITH DIGITAL COMPUTER PROGRAMS", PG.44.
C  INPUTS:
C    LR= LENGTH OF FILTER F
C    R= VECTOR OF AUTOCORRELATION COEFS
C    G= VECTOR OF RIGHT-HAND SIDE COEFFS IN THE NORMAL EQUATIONS
C  OUTPUTS:
C    F= VECTOR OF FILTER COEFFS ,WHICH IS THE SOLUTION TO THE NORMAL EQU
                                  **ATIONS.
C    A= PREDICTION ERROR OPERATOR VECTOR
C
      DIMENSION R(1),G(1),F(1),A(1)
C
      V= R(1)
      B= R(2)
      A(1)= 1.
      F(1)= G(1)/V
      Q= F(1)*R(2)
      IF(LR.EQ.1) RETURN
      DO 4 L=2,LR
      A(L)= -B/V
      IF(L.EQ.2) GO TO 2
      L1= (L-2)/2
      L2= L1+1
      IF(L2.LT.2) GO TO 5
      DO 1 J=2,L2
      MULB= A(J)
      K= L-J+1
      A(J)= A(J)+A(L)*A(K)
1     A(K)= A(K)+A(L)*MULB
5     IF(2*L1.EQ.L-2) GO TO 2
      A(L2+1)= A(L2+1)+A(L)*A(L2+1)
2     V= V+A(L)*B
      F(L)= (G(L)-Q)/V
      L3= L-1
      DO 3 J=1,L3
      K= L-J+1
3     F(J)= F(J)+F(L)*A(K)
      IF(L.EQ.LR) RETURN
      B=0.
      Q=0.
      DO 4 I=1,L
      K= L-I+2
      B= B+A(I)*R(K)
4     Q= Q+F(I)*R(K)
      RETURN
      END
```



```
      SUBROUTINE LTRANS(B,ITYP)
C
C  ROUTINE TO LINEARLY TRANSFORM (AND INVERSE TRANSFORM)
C  THE PITCH PREDICTOR TAP COEFFICIENTS FOR
C  THE 3-TAP CASE.
C    ITYP= FALSE, FORWARD TRANSFORM
C          TRUE, INVERSE TRANSFORM
C
      DIMENSION B(1),PSI(3)
      DO 1 I=1,3
1     PSI(I)= B(I)
      IF(ITYP) GO TO 10
C  FORWARD TRANSFORM
      B(1)= PSI(1)+PSI(2)+PSI(3)
      B(2)= PSI(1)-2.*PSI(2)+PSI(3)
      B(3)= PSI(1)-PSI(3)
      GO TO 100
C  INVERSE TRANSFORM
10    B(1)= (2.*PSI(1)+PSI(2)+3.*PSI(3))/6.
      B(2)= (PSI(1)-PSI(2))/3.
      B(3)= (2.*PSI(1)+PSI(2)-3.*PSI(3))/6.
100   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DELTAG(X,MAXX,ETOT,NST,EDEV)
C
C  COMPUTES DELTA GAINS
C
C   INPUTS:
C       X       VECTOR OF INPUT SAMPLES
C       MAXX    MAXIMUM DIMENSION OF X
C       ETOT    TOTAL MEAN SQUARED FRAME ENERGY
C       NST     INDEX OF THE START SAMPLE IN X FROM
C               WHICH THE SECTION ENERGIES ARE TO BE COMPUTED.
C   OUTPUT:
C       EDEV    VECTOR OF DELTA GAINS
C
      DIMENSION X(MAXX),EDEV(1)
C
      COMMON /NSEC/NSNSEC,NSMSEC
C
      N1=NST
      DO 10 I=1,NSNSEC
      N2=N1+NSMSEC-1
      EDEV(I)=ENERGY(X,MAXX,N1,N2)
      EDEV(I)=EDEV(I)/ETOT
      N1=N2+1
10    CONTINUE
      RETURN
      END
```



```
      SUBROUTINE QSCALE(G,ETOT,NST,EDEV)
C
C  COMPUTES THE QUANTIZER SCALE FACTORS
C
C  INPUTS:
C       ETOT    TOTAL MEAN SQUARE FRAME ENERGY
C       EDEV    VECTOR OF DELTA GAINS
C       NST     INDEX OF THE START LOCATION IN THE
C               ARRAY G WHERE THE SCALE FACTORS ARE STORED.
C  OUTPUT:
C       G       ARRAY OF QUANTIZER SCALE FACTORS
C               NOTE THAT THE ARGUMENT OF G IS THE SAMPLE NUMBER,
C               I.E. THE SCALE FACTOR FOR INDIVIDUAL SAMPLES ARE ,
C               IN GENERAL , NOT THE SAME.
C
      DIMENSION G(1),EDEV(1)
C
      COMMON /NSEC/NSNSEC,NSMSEC
C
      N1=NST
      DO 20 I=1,NSNSEC
      N2=N1+NSMSEC-1
      E=ETOT*EDEV(I)
      DO 10 J=N1,N2
      G(J)=E
10    CONTINUE
      N1=N2+1
20    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE STBLZE(B,NTAPS,B0)
C
C  CHECK FOR STABILITY OF THE 3-TAP PITCH FILTER
C  AND REPLACE IT BY 1-TAP FILTER WHENEVER INSTABILITY
C  IS DECLARED.
C
C   INPUTS:
C        B       PITCH PREDICTOR TAPS
C                (REPLACED BY 1-TAP FILTER COEFFS
C                IF INSTABILITY DECLARE, OTHERWISE
C                UNCHANGED AND RETURNED TO CALLING PROGRAM)
C        NTAPS   NUMBER OF PITCH PREDICTOR TAPS
C        B0      PITCH PREDICTOR TAP FOR 1-TAP CASE
C
C
      DIMENSION B(1)
      COMMON/TSCR/TSCR(3)
C
C
      DO 10 I=1,NTAPS
10    TSCR(I)=B(I)
C
      CALL LTRANS(TSCR,1)
C
C  THE STABILITY - CHECK RULES THAT FOLLOW
C  HAVE BEEN EMPIRICALLY OBTAINED
      DO 20 I=1,NTAPS
20    IF(TSCR(I).LT.-1.0) GO TO 25
      IF(TSCR(2).GT.2.0) GO TO 25
      IF(TSCR(3).GT.1.0) GO TO 25
C  NO INSTABILITY
      RETURN
C  INSTABILITY DECLARED ; REPLACE WITH 1-TAP FILTER
25    NTAPM=(NTAPS+1)/2
      CALL SETZR(B,NTAPS)
      B(NTAPM)=B0
      RETURN
      END
^L
```



```
      SUBROUTINE PZNS(A,ANS,PZFAC)
C
C  COMPUTES COEFFICIENTS OF THE NUMERATOR
C  OF THE NOISE SHAPING FILTER.
C
C  INPUTS:
C        A       SPECTRAL PREDICTOR COEFFS (NPOLE IN NUMBER)
C        PZFAC   CONSTANT
C  OUTPUT:
C        ANS     NUMERATOR COEFFS OF NOISE SHAPING FILTER
C
C
      DIMENSION A(1),ANS(1)
C
      COMMON/NPOLE/NPOLE
C
      BWFACT=1.0
      DO 2154 I=1,NPOLE
      BWFACT= BWFACT*PZFAC
2154  ANS(I)= A(I)*BWFACT
      RETURN
      END
^L
```

```
      SUBROUTINE APC(S,GSEC,ANS,DA,BR,MPTCNR,N2S,WO,IR,VM,MM,Q,QI
     1,NTAPS,IQ,KPPF,KNSF)
C
C  APC LOOP WITH POLE-ZERO NOISE SHAPING.  USES P-S
C  PREDICTION SEQUENCE AND PREDICTIVE FEEDBACK.  PITCH
C  PREDICTION AND/OR NOISE SHAPING CAN BE DELETED FROM
C  THE LOOP BY ZEROING THE FLAGS DESCRIBED BELOW.
C  INPUTS:
C       S      = INPUT SAMPLES (START AT N2S)
C       GSEC   = QUANTIZER SCALE FACTOR ARRAY (START AT N2S)
C       ANS    = FILTER COEFFICIENTS USED TO IMPLEMENT THE
C                NUMERATOR (ZEROES) OF THE NOISE SHAPING FILTER
C                (NPOLE TOTAL).
C       DA     = SPECTRAL PREDICTOR COEFFICIENTS (NPOLE TOTAL)
C       BR     = PITCH PREDICTOR COEFFICIENTS (NTAPS TOTAL)
C       MPTCNR = PITCH LAG IN # OF SAMPLES
C       KPPF   =  PITCH PREDICTION FLAG(TRUE=ON,FALSE=OFF)
C       KNSF   = NOISE SHAPING FLAG (TRUE=ON,FALSE=OFF)
C       IQ     = FLAG FOR RESIDUAL SAMPLE QUANTIZATION
C                (TRUE(-1)=ON, FALSE(0)= OFF)
C  OUTPUTS:
C       IR  = CODED RESIDUAL SAMPLES
C       WO,VM,MM,Q,QI=INTERMEDIATE ARRAYS OF APC LOOP
C                     (ALL SAME SIZE AS S)
C                     SEE REAL TIME SPECIFICATIONS
C                     OF BBN 16KB/S APC CODER FOR
C                     DESCRIPTION OF ARRAY FUNCTIONS
C
      DIMENSION S(1),GSEC(1),ANS(1),DA(1),BR(1)
      DIMENSION WO(1),IR(1),VM(1),MR(1),Q(1),QI(1)
C
      COMMON/FRMSIZ/NFRAME,NdIND
      COMMON/NPOLE/NPOLE
      COMMON/RESCOD/NLEVR,XRES(9),YRES(8)
C
      II=N2S-1
C
C  TOP OF APC LOOP
      DO 2600 N=1,NFRAME
      II=II+1
C  NOISE SHAPING
      QP=0.
      QP1=0.
      IF(.NOT.KNSF) GO TO 2630
      CALL PRED(QI,QP,II,ANS,NPOLE,0)
      CALL PRED(QI,QP1,II,DA,NPOLE,0)
2630  CONTINUE
C  SPECTRAL FILTER
      CALL PRED(VM,VMP,II,DA,NPOLE,0)
C  PITCH PREDICTOR
      MMP=0.
      IF(.NOT.KPPF) GO TO 2650
      CALL PRED(MM,MMP,II,BR,NTAPS,MPTCNR)
```



```
2650  CONTINUE
C  COMPUTE WO
      WO=S(II)+QP-QP1+MMP+VMP
C  ADAPTIVE QUANTIZER
      WO(II)=WO/GSEC(II)
C  QUANTIZE
      IF(.NOT.IQ) GO TO 2660
      CALL CODU(WO(II),XRES,NLEVR,IR(N))
      CALL DCOD(IR(N),YRES,UM)
      GO TO 2665
C  NO CODING
2660  UM=WO(II)
2665  CONTINUE
C
      MM=UM*GSEC(II)
C  COMPUTE Q AND Q1
      Q(II)=MM-VM
      QI(II)=Q(II)-QP1
C
C  COMPUTE VM
      VM(II)=MM-VMP
C
C  COMPUTE RM
      RM(II)=VM(II)-RMP
C  END OF LOOP
2600  CONTINUE
      RETURN
      END
^L
```

```
      SUBROUTINE HPCOR(RSCR,NPWFC,CWF,WPLAN)
C
C  APLIES HIGH FREQUENCY CORRECTION TO AUTOCORRELATION
C  COEFFICIENTS.
C  ARGUMENTS:
C     RSCR=     VECTOR CONTAINING INPUT AUTOCORRELATION
C               COEFFICIENTS.(ALSO CONTAINS HIGH FREQUENCY CORRECTED
C                OUTPUT AUTOCORRELATIONS)
C     NPWFC=    NUMBER OF POLES USED FOR EMIN COMPUTATION.
C     CWF=      AUTORRELATION COEFFICIENTS FOR HIGHPASS
C               FILTERED WHITE NOISE.
C     WPLAN=    CONSTANT (.01 < WPLAN < .1). THAT DETERMINES
C                THE EXTENT OF CORRECTION.
C
        DIMENSION RSCR(1),CWF(1)
        DIMENSION SCR1(10),SCR2(10),SCR3(10)
C
        CALL ALPCOR(NPWFC,1,RSCR,SCR1,SCR2,V,SCR3)
        EMIN=RSCR(1)*V
        DO 10 I=1,3
10      RSCR(I)=RSCR(I)+WPLAN*EMIN*CWF(I)
        RETURN
        END
^L
```


END
DATE
FILMED
4-81
DTIC